МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»

О.В. КОЗЕЛКОВ, И.В. ЛОМАКИН

КУРСОВОЙ ПРОЕКТ

методические указания к выполнению курсового проекта по дисциплине МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА В МЕХАТРОНИКЕ И РОБОТОТЕХНИКЕ

Казань 2020

УДК 004.431.4
ББК 32.973.26-018.2
К59

Рецензенты:

доктор технических наук, доцент ФГБОУ ВО
«Казанский национальный исследовательский технологический университет»
В.Г. Макаров;
кандидат технических наук, доцент ФГБОУ ВО
«Казанский государственный энергетический университет»
В.В. Максимов

К59 **Козелков О.В., Ломакин И.В.**

Микропроцессорные устройства в мехатронике и робототехнике: методические указания по выполнению курсового проекта/ О.В. Козелков, И.В. Ломакин. – Казань: Казан. гос. энерг. ун-т, 2020. – 95 с. : ил.

Способствует закреплению знаний по архитектуре, функционированию и технологии программирования микроконтроллеров на примере микроконтроллеров AT90S4434/8535 семейства AVR, а также приобретению навыков программирования и отладки программ на языке ассемблера.

Предназначен для студентов очной формы обучения, обучающихся по образовательной программе «Мехатроника и робототехника» направления подготовки бакалавров 15.03.06 Мехатроника.

УДК 004.431.1
ББК 32.973.26-018.2

ВВЕДЕНИЕ

Лабораторный практикум по дисциплине «Микропроцессорные средства в мехатронике и робототехнике» предназначен для закрепления знаний по архитектуре и функционированию основных модулей микроконтроллеров на примере микроконтроллеров AT90S4434/8535 семейства AVR, а также приобретения навыков программирования и отладки программ на языке ассемблера.

Для удобства обучающихся в первой части работы дано описание основных элементов языка ассемблера семейства AVR: системы счисления, машинное представление данных и команд. Кроме того, приведены директивы ассемблера для AVR и система команд процессора AT90S4434/8535 и некоторые дополнительные сведения к ним. Подробно рассмотрены особенности программирования микропроцессорных систем на языке ассемблера, организации интерфейса с устройствами ввода/вывода и хранения информации, указаны возможные проблемы при вычислениях и способы их устранения.

Один из разделов практикума описывает порядок создания, компиляции и отладки программ на этапе разработки с помощью программы Avr Studio. Для записи программы в микроконтроллер используется программа New_SP.

Лабораторный практикум предназначен для студентов, обучающихся по образовательной программе «Мехатроника» направления подготовки бакалавров 15.03.06 Мехатроника и робототехника, и направлен на формирование у обучающегося способности и готовности к выбору оптимального метода и разработке программ экспериментальных исследований, проведению измерений с выбором технических средств и обработкой результатов.

ЧАСТЬ I. ЛАБОРАТОРНЫЙ КОМПЛЕКС

1. МИКРОКОНТРОЛЛЕРЫ AT90S4434/8535 СЕМЕЙСТВА AVR

1.1. Структура микроконтроллеров AVR

Упрощенно модульную структуру микроконтроллера семейства AVR можно представить в следующем виде (рис. 1).

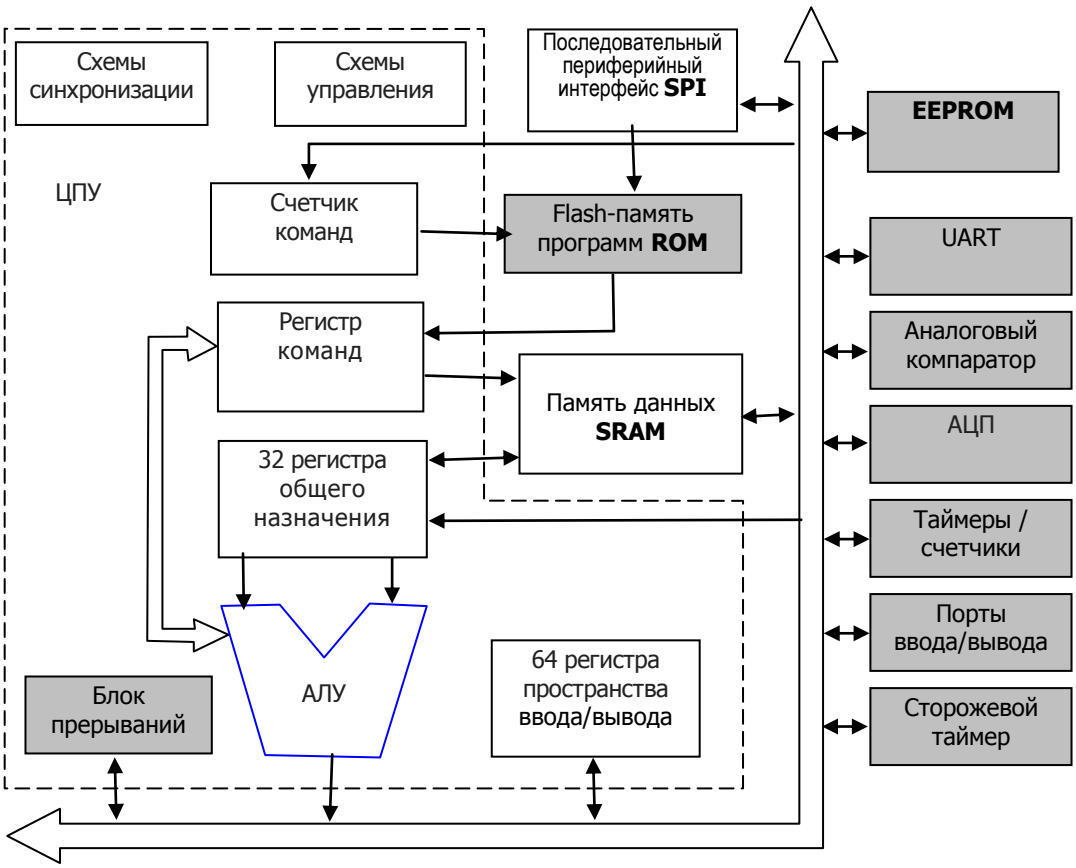


Рис. 1. Структура микроконтроллера семейства AVR

AVR представляет собой 8-разрядный RISC-микроконтроллер, имеющий быстрое процессорное ядро, внутрисистемно программируемую Flash-память программ (ROM) емкостью 8 Кбайт, EEPROM-память данных емкостью 512 байт, память данных SRAM (512 байт), порты ввода/вывода и интерфейсные схемы.

В микроконтроллерах AVR использованы принципы гарвардской архитектуры – отдельные память и шины для программ и данных.

При работе с памятью программ используется одноуровневый конвейер – в то время, как одна команда выполняется, следующая команда выбирается из памяти программ. Такой прием позволяет выполнять команду в каждом тактовом цикле.

1.2. Представление данных на языке ассемблера

Для освоения программирования на ассемблере необходимо познакомиться с двоичными и шестнадцатеричными числами, так как без понимания того, как хранятся данные в памяти компьютера, трудно использовать логические и битовые операции.

Двоичная система счисления. В электрических цепях микропроцессорных систем напряжение может принимать два значения – низкого и высокого уровня (нет сигнала и есть сигнал). Этим двум значения поставили в соответствие две цифры – ноль и единицу. Именно эти две цифры и используются в двоичной системе счисления, а вместо степеней десяти, как в обычной десятичной системе, – степени двойки. Для перевода двоичного числа в десятичное необходимо сложить двойки в степенях, соответствующих позициям, где в двоичном числе стоят единицы. Например:

$$10010111b =$$
$$= 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 =$$
$$= 128 + 16 + 4 + 2 + 1 = 151.$$

Для перевода десятичного числа в двоичное можно, например, разделить его на два, записывая остатки справа налево (табл. 1).

Чтобы отличать двоичные числа от десятичных, в тексте в конце каждого двоичного числа ставится буква **b**.

Таблица 1

Перевод десятичного числа в двоичное

Частное	Остаток
151 / 2	1
75 / 2	1
37 / 2	1
18 / 2	0
9 / 2	1
4 / 2	0
2 / 2	0
1 / 2	1
Результат	10010111b

Биты, байты и слова. *Битом* называется минимальная единица информации. Бит либо есть, либо его нет, т.е. можно сказать, что он принимает два значения – ноль и единица, «да» и «нет» и т.п. Важно лишь то, что бит имеет только два значения. Очевидно, что один разряд двоичного числа несет один бит информации.

Единица информации размером восемь бит называется *байт*. Байт – это минимальный объем данных, который реально может использовать компьютерная программа. Даже для изменения одного бита в памяти приходится считывать байт, содержащий его. Биты в байте нумеруются справа налево, от нуля до семи (рис. 2). Нулевой бит часто называют младшим битом, седьмой – старшим.

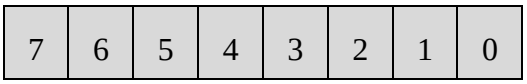


Рис. 2. Байт

Так как в байте всего восемь бит, то он может принимать до $2^8 = 256$ различных значений. Байт используется для представления целых чисел от 0 до 255, целых чисел со знаком от –128 до +127 или любых переменных, принимающих менее 256 значений.

Следующий по размеру базовый тип данных – **слово**. Размер одного слова – два байта. Биты с нулевого по седьмой составляют младший байт слова, а биты с восьмого по пятнадцатый – старший. В слове содержится шестнадцать бит, а значит, оно может принимать до $2^{16} = 65536$ различных значений. Два слова подряд образуют двойное слово (32 бит), а два двойных слова – одно учетверенное слово (64 бит).

Шестнадцатеричная система счисления. Главное неудобство двоичной системы счисления – это размеры чисел, с которыми приходится обращаться. На практике с двоичными числами работают, только если необходимо следить за значениями отдельных битов, а когда размеры переменных превышают четыре бита, используется шестнадцатеричная система. Она хороша тем, что компактнее десятичной, и тем, что перевод в двоичную систему и обратно происходит очень легко. В шестнадцатеричной системе используются шестнадцать знаков: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Каждая позиция цифры в числе соответствует степени числа шестнадцать. Например:

$$97h = 9 \times 16^1 + 7 \times 16^0 = 9 \times 16 + 7 = 151.$$

Перевод в двоичную систему и обратно осуществляется просто: вместо каждой шестнадцатеричной цифры подставляют соответствующее четырехзначное двоичное число, и наоборот:

$$97h = (9h = 1001b, 7h = 0111b) = 10010111b.$$

Чтобы отличать шестнадцатеричные числа от десятичных, в тексте в конце каждого шестнадцатеричного числа ставится буква **h**.

Иногда в ассемблерных программах используется восьмеричная система счисления. В восьмеричной системе используются восемь знаков: 0, 1, 2, 3, 4, 5, 6, 7. Каждая позиция цифры в числе соответствует степени числа восемь. Например:

$$0227h = 2 \cdot 8^2 + 2 \cdot 8^1 + 7 \cdot 8^0 = 2 \cdot 64 + 2 \cdot 8 + 7 = 128 + 16 + 7 = 151.$$

Перевод в двоичную систему и обратно осуществляется просто: вместо каждой восьмеричной цифры подставляют соответствующее трехзначное двоичное число, и наоборот:

$$0227h = (2 = 010b, 2 = 010b, 7 = 111b) = 10010111b.$$

Чтобы отличать восьмеричные числа от десятичных, в тексте перед каждым восьмеричным числом ставится **0**.

Для того чтобы в программах при записи чисел не перепутать системы счисления, а числа, начинающиеся с A, B, C, D, E, F, – с переменными, в каждом ассемблере принимают свои обозначения.

В ассемблере для AVR приняты следующие форматы операндов:

– десятичный (по умолчанию): 10, 255;

– шестнадцатеричный (два способа): **0x**0a, **$**0a;

– двоичный: **0b**00001010, 0b11111111;

– восьмеричный (впереди ноль): **0**10, **0**77.

В табл. 2 дано соответствие десятичных, двоичных и шестнадцатеричных чисел от нуля до шестнадцати.

Таблица 2

Соответствие десятичных, двоичных и шестнадцатеричных чисел

Десятичное	Двоичное	Шестнадцатеричное	Восьмеричное
0	0b0000	$00	00
1	0b0001	$01	01
2	0b0010	$02	02
3	0b0011	$03	03
4	0b0100	$04	04
5	0b0101	$05	05
6	0b0110	$06	06
7	0b0111	$07	07
8	0b1000	$08	010
9	0b1001	$09	011
10	0b1010	$0A	012
11	0b1011	$0B	013
12	0b1100	$0C	014
13	0b1101	$0D	015
14	0b1110	$0E	016
15	0b1111	$0F	017
16	0b10000	$10	020

1.3. Адресное пространство микроконтроллеров AVR

В микроконтроллерах AVR все регистры и ***память данных*** располагаются в одном адресном пространстве. Это означает, что ***память данных*** совмещена с регистрами. Такой подход называется «отображением ресурсов ***МК*** на память».

Памятью программ является внутрисистемно программируемая Flash-память объемом 8 Кбайт. Поскольку все команды имеют формат одного или двух 16-разрядных слоев, то и память программ имеет организацию 4К × 16. Flash-память обеспечивает не менее 1000 циклов стирания/записи.

Счетчик программ AT90S8535 (PC) является 12-разрядным в зависимости от объема памяти программ (4096). Адреса памяти программ – от $0000 до $FFF.

Память данных AVR микроконтроллера AT90S8535 состоит из:

– 32-х 8-разрядных регистров общего назначения;

– 64-х 8-разрядных регистров пространства памяти I/O;

– статического ОЗУ (СОЗУ, SRAM) объемом 512 байт (512 × 8);

– блока энергонезависимой памяти данных EEPROM (ЭСППЗУ) объемом 512 байт (512 × 8), доступного программе микроконтроллера непосредственно в ходе ее выполнения.

EEPROM обычно используется для хранения промежуточных данных, констант, таблиц перекодировок, калибровочных коэффициентов и т.п. Эта память может быть загружена извне как через SPI-интерфейс, так и с помощью обычного программатора. EEPROM имеет собственный набор адресов от $0000 до $1FF.

Адресное пространство представлено на рис. 3.

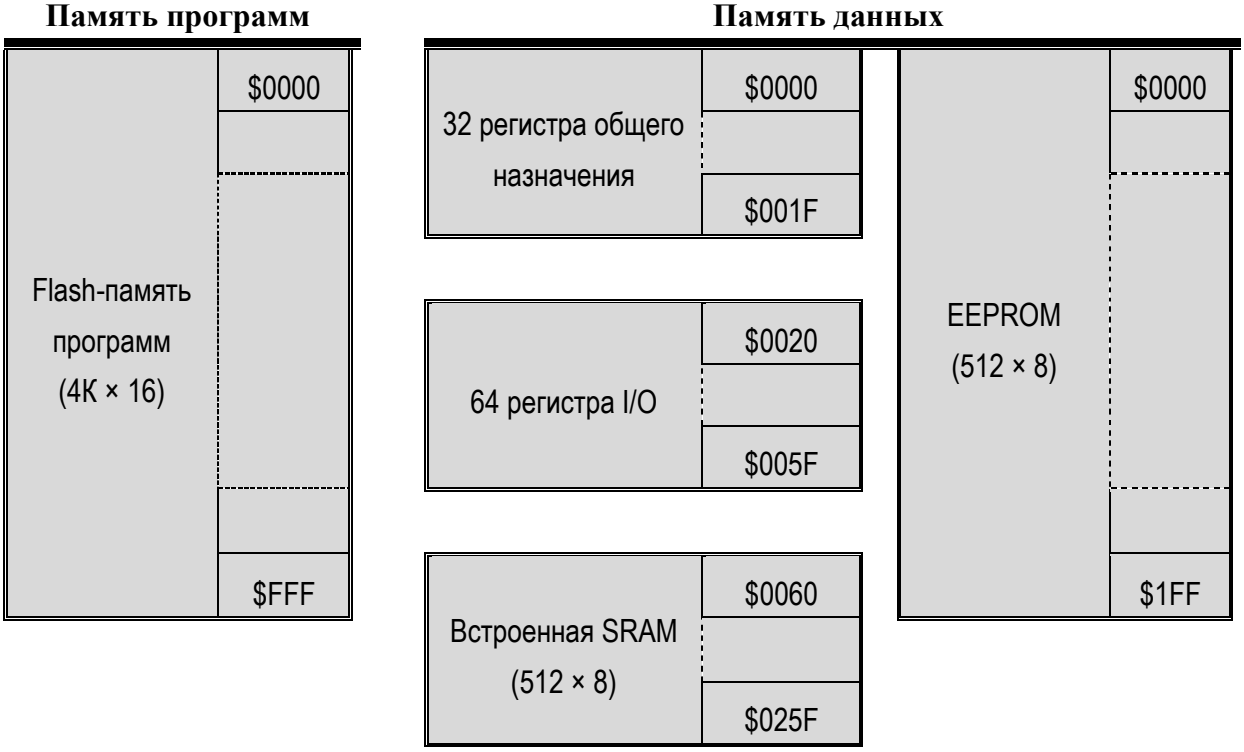


Рис. 3. Конфигурация памяти микроконтроллера AT90S8535

Первые 96 адресов занимают регистровый файл и пространство регистров ввода/вывода. В следующих 512 адресах размещается встроенная SRAM; 32 регистра общего назначения и 64 регистра I/O, кроме адресов в общем пространстве памяти данных, имеют и свои собственные адреса, по которым к ним можно обращаться с помощью специальных команд (рис. 4).

Дополнительные обозначения		Собственные адреса (обозначения)	Адреса в общем пространстве памяти данных
Файл регистров общего назначения		R0	$0000
		…	…
		R25	$0019
	X - регистр	R26	$001A
		R27	$001B
	Y - регистр	R28	$001C
		R29	$001D
	Z - регистр	R30	$001E
		R31	$001F
Файл регистров пространства ввода - вывода		$00	$0020
		…	…
		$3F	$005F
Встроенная SRAM			$0060
			…
			$025F

Рис. 4. Организация SRAM микроконтроллера AT90S8535

Всеми режимами адресации перекрывается все адресное пространство данных, включая 32 регистра общего назначения, 64 регистра ввода/вывода и 512 байт внутренних данных SRAM в AT90S8535.

В табл. 3 показаны адреса и наименования всех регистров пространства ввода/вывода.

Таблица 3

Пространство ввода/вывода (I/O) микроконтроллеров AT90S4434/8535

Адрес I/O (адрес SRAM)	Обозначение	Функция
$3B($5B)	GIMSK	General Interrupt MaSK register
$3A($5A)	GIFR	General Interrupt Flag Register
$39($59)	TIMSK	Timer/Counter Interrupt MaSK register
$38($58)	TIFR	Timer/Counter Flag Register
$35($5)	MCUCR	MCU general Control Register
$34($45)	MCUSR	MCU general Status Register
$33($53)	TCCR0	Timer/CounterO Control Register
$32($52)	TCNT0	Timer/CounterO 8-bit
$2F($4F)	TCCR1A	Timer/Counteri Control Register A
$2E($4E)	TCCR1B	Timer/Counteri Control Register B
$2D($4D)	TCNT1H	Timer/Counteri, ст. байт
$2C($4C)	TCNT1L	Timer/CounteM, мл. байт
$2B($4B)	OCR1AH	Timer/Counteri Output Compare Register A, ст. байт
$2A($4A)	OCR1AL	Timer/Counteri Output Compare Register A, мл. байт
$29($49)	OCR1BH	Timer/Counteri Output Compare Register B, ст. байт
$28($48)	OCR1BL	Timer/Counteri Output Compare Register B, мл. байт
$27($47)	OCR1H	T/C Input Capture Register, ст. байт
$26($46)	OCR1L	T/C Input Capture Register, мл. байт
$25($45)	TCCR2	Timer/Counter2 Control Register
$24($44)	TCNT2	Timer/Counter2 8-bit
$23($43)	OCR2	Timer/Counter2 Output Compare Register
$22($42)	ASSR	Asynchronous Mode Status Register
$1B($3B)	PORTA	Data Register, Port A
$1A($3A)	DDRA	Data Direction Register, Port A
$19($39)	PINA	Input Pins, Port A
$18($38)	PORTB	Data Register, Port B
$17($37)	DDRB	Data Direction Register, Port B
$16($36)	PINB	Input Pins, Port B
$15($35)	PORTC	Data Register, Port C
$14($34)	DDRC	Data Direction Register, Port C
$13($33)	PINC	Input Pins, Port C
$12($32)	PORTD	Data Register, Port D
$11($31)	DDRD	Data Direction Register, Port D
$10($30)	PIND	Input Pins, Port D

2. ОПИСАНИЕ ЛАБОРАТОРНОГО КОМПЛЕКСА

2.1. Назначение и состав комплекса

Лабораторный комплекс «Микроконтроллеры и автоматизация» предназначен для обучения студентов программированию интегральных микроконтроллеров (однокристальных микро-ЭВМ).

Комплекс предназначен для изучения программирования микроконтроллеров **AT90S8535** и **AT90S4434** семейства **AVR**, выпускаемых фирмой Atmel.

Основой комплекса является ПЭВМ, которая в состав стенда не входит, а используется ПЭВМ, имеющаяся на месте проведения занятий.

Комплекс включает в себя следующие изделия и блоки:

блок связи с ЭВМ (БС) . . . . . 1 шт.;
блок питания комплекса (БП) . . . . . 1 шт.;
клавиатура рабочего места . . . . . 8 шт.;
блок управления рабочего места (БУ) . . . . . 8 шт.;
кабели рабочих мест . . . . . 8 шт.;
кабель сетевой блока питания . . . . . 1 шт.;
кабель «Блок питания – блок связи» . . . . . 1 шт.;
кабель соединительный «Блок связи – ПЭВМ (port com2)» . . . . . 1 шт.

В лабораторном комплексе предусматривается использование программирования Flash-памяти в режиме низковольтного последовательного программирования. Этот режим является обычным способом загрузки программы и данных в микроконтроллер, находящийся непосредственно в системе пользователя. Число перепрограммирований Flash-памяти – 1 000, а EEPROM – 100 000.

Комплекс позволяет организовать от одного до восьми рабочих мест. На каждом рабочем месте располагаются клавиатура и блок управления.

Клавиатура рабочего места – покупная, от персональных компьютеров типа IBM и подключается к блоку управления через свой штатный разъем.

2.2. Блок управления

Блок управления включает в себя две платы: плату управления и плату индикации. Упрощенный вид блока управления представлен на рис. 5.

Аппараты платы управления выведены на лицевую панель в верхней части блока управления, а элементы платы индикации – в нижней части.

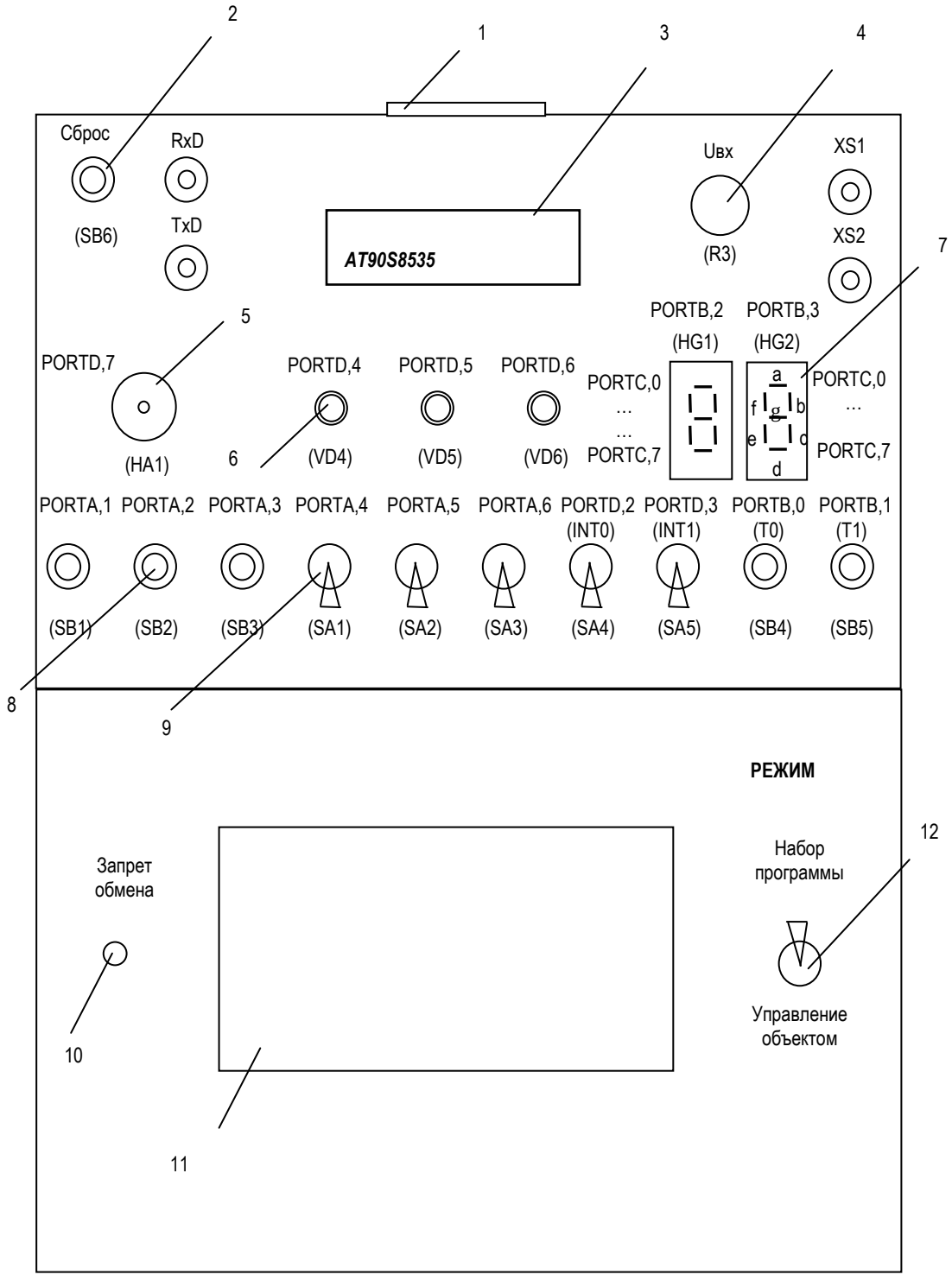


Рис. 5. Упрощенный вид блока управления

Пять кнопок *8* и пять тумблеров *9* используются в качестве источников входных сигналов. Кнопка *2* «Сброс» обеспечивает аппаратный сброс микроконтроллера.

Микроконтроллер *3* установлен на плате в 40-контактной панельке. Программирование его осуществляется непосредственно на месте его установки.

На плате управления располагается коммутационная и сигнальная аппаратура, которая необходима для подачи входных сигналов на микроконтроллер и отображения выходных сигналов микроконтроллера.

На лабораторных работах предусматривается ввод команд с блока управления в виде:

– нажатия-отпускания кнопок;

– включения-выключения тумблеров.

Наблюдение реакции на эти команды осуществляется:

– по загоранию-погасанию светодиодов;

– по включению или изменению тона звукогенератора;

– по индикации информации на семисегментных индикаторах.

При решении каждой задачи необходимо выбрать, какие кнопки или ключи будут использованы, т.е. определить их адреса, и разобраться с адресацией индикаторов и принципами управления ими.

В качестве органов управления использовано пять кнопок (SB1–SB6) и пять тумблеров (SA1–SA5). Кнопки SB1–SB5 и тумблеры SA1–SA5 подключены однотипно. При не нажатой кнопке и отключенном тумблере (нижнее положение тумблера) на соответствующие выводы микроконт-роллера подается уровень логического нуля, а при нажатой кнопке и включенном тумблере – уровень логической единицы.

Для борьбы с влиянием дребезга контактов дополнительно на входах запросов внешнего прерывания INT0, INT1 и на входах таймеров/счетчиков T0 и T1 установлены RC-цепочки.

Вращением движка потенциометра *4* «Uвх» обеспечивается изменение значения напряжения на входе аналогового преобразователя.

Гнезда XS1 и XS2 служат для подключения измерительного прибора для контроля напряжения, подаваемого на вход АЦП. Гнезда RxD и TxD служат для наблюдения по осциллографу сигналов RxD и TxD при работе последовательного порта UART.

Через вилку разъема XP5 осуществляется связь блока связи с ЭВМ. Через эту вилку связываются с портом COM2 ПЭВМ последовательные порты микроконтроллера SPI и UART, а также подается напряжение питания блока управления.

Сброс микроконтроллера осуществляется внешним сигналом. В блоке управления рабочего места эту роль выполняет кнопка SB6 «Сброс». При ее нажатии сигнал низкого уровня подается на вывод RESET. Для получения надежного сброса эту кнопку следует удерживать в нажатом состоянии 1–3 с.

При включении питания или при нажатии кнопки «Сброс» формируется прерывание по вектору $000 с наивысшим приоритетом.

По этому адресу должна располагаться команда перехода **rjmp** к началу программы пользователя. При любом сбое в работе программы микроконтроллера прибегайте к услугам кнопки «Сброс».

В качестве выходных элементов используются три светодиода (зеленый, желтый, красный) VD4–VD6, звукоизлучатель («пищалка») HA1 и два семисегментных индикатора HG1 и HG2.

В блоке управления используются семисегментные индикаторы HG1 и HG2 с общим катодом. Для замыкания катодов индикаторов на общий провод служат транзисторы VT1 и VT2, которые управляются соответственно битами PB2 и PB3 порта B. Транзисторы VT1 и VT2 при соответствующей программе обеспечивают динамическую индикацию информации на семисегментных индикаторах, т.е. вывод различной информации на оба индикатора. В этом случае поочередно зажигается то первый, то второй индикатор. При достаточно большой частоте переключений создается иллюзия одновременного непрерывного свечения индикаторов.

Для зажигания информации на первом индикаторе необходимо установить в «1» бит PB2 и записать в порт C код выводимого первого символа. Затем, через некоторую программно реализуемую задержку времени, нужно сбросить бит PB2, установить в «1» бит PB3 и подать на порт C код второго выводимого символа. Через задержку времени нужно сбросить бит PB3, установить в «1» бит PB2 и т.д.

Адресация входных и выходных сигналов блока управления очевидна из рис. 5. На лицевой панели пульта управления рядом с каждым элементом указана его адресация. В скобках указаны позиционные обозначения элементов в принципиальной схеме блока управления. У тумблеров SA4, SA5 и кнопок SB4, SB5 в скобках дополнительно указано функциональное обозначение входов микроконтроллера, на которые воздействуют указанные тумблеры и кнопки.

В нижней части блока управления располагается ЖКИ *11*, переключатель режима работы *12* и светодиод *10* индикации работы ЖКИ. Переключатель режима работы имеет два положения. Первое положение «Набор программы» обеспечивает связь ЖКИ с ПЭВМ. Во втором режиме «Управление объектом» ЖКИ отключается от последовательного порта ПЭВМ, а к последовательному порту ПЭВМ подключается последовательный порт UART изучаемого микроконтроллера для управления виртуальным объектом автоматизации, воспроизводимым на экране монитора ПЭВМ.

3. ДИРЕКТИВЫ АССЕМБЛЕРА

Синтаксис всех директив следующий:

директива [выражение]

Перед директивой должна стоять точка, иначе ассемблер воспринимает это как метку.

3.1. Директивы организации сегментов

.DSEG

Директива DSEG указывает на начало сегмента данных (Data Segment). Директива показывает, что следующие за ней данные в виде меток, имен переменных или констант должны быть расположены в ОЗУ (регистровом файле, файле I/O, SRAM). Обычно сегмент данных состоит лишь из директив BYTE и меток. Директива ORG может использоваться для размещения переменных в нужной области ОЗУ.

Программа может содержать несколько сегментов данных, которые потом будут собраны в один при ассемблировании.

.ESEG

Директива ESEG указывает на начало сегмента EEPROM (EEPROM Segment). Директива показывает, что следующие за ней данные в виде меток или имен констант должны быть расположены в EEPROM (ЭСППЗУ). Обычно сегмент EEPROM состоит из DB и DW директив и меток. Сегмент EEPROM памяти имеет свой собственный счетчик. Директива ORG может использоваться для размещения переменных в нужной области EEPROM.

Программа может содержать несколько EEPROM-сегментов, которые будут собраны в один сегмент при ассемблировании.

.CSEG

Директива CSEG указывает на начало сегмента кодов (Code segmen). Директива показывает, что следующие за ней данные в виде машинных команд, меток или имен констант должны быть расположены во Flash-памяти программ. Директива ORG может использоваться для размещения команд и констант в нужной области памяти программ.

Программа может иметь несколько кодовых сегментов, которые будут объединены в один при ассемблировании.

3.2. Директивы счетчика текущего адреса

.ORG ***адрес***

Директива ORG присваивает значения локальным счетчикам. Используется только совместно с директивами .CSEG, .DSEG, .ESEG.

В одном сегменте может использоваться несколько раз.

3.3. Директивы определения данных

label: **.DB** ***список выражений***

Директива DB резервирует ресурсы памяти и определяет значения байта(ов) в программной памяти или в EEPROM. Директиве должна предшествовать метка *label* или директива ORG.

DB задает список выражений и должна содержать по крайней мере одно выражение. Размещать директиву следует в сегменте кодов или в EEPROM-сегменте.

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.

label: **.DW** ***список выражений***

Директива DW резервирует ресурсы памяти и определяет значения слов в программной памяти или в EEPROM. Директиве должна предшествовать метка или директива ORG.

DW задает список выражений и должна содержать по крайней мере одно выражение. Размещать директиву следует в сегменте кодов или в EEPROM-сегменте.

Список выражений представляет собой последовательность выражений, разделенных запятыми. Каждое выражение должно быть величиной между 32768 и 65535.

label_var: **.BYTE** ***m***

Директива BYTE резервирует *m* байт под переменную *label_var* в области ОЗУ. Размещать директиву следует в сегменте данных.

3.4. Директивы присваивания

.EQU *имя_к=выражение*

Директива EQU присваивает символическое *имя_к* ячейке ОЗУ, адрес которой определяется *выражением*. Это имя может быть использовано в других выражениях вместо адреса. Значение этого имени нельзя изменить или переопределить.

.SET *имя_к=выражение*

Директива SET присваивает символическое *имя_к* ячейке ОЗУ, адрес которой определяется *выражением*. Это имя может быть использовано в других выражениях вместо адреса. Значение этой метки можно изменить или переопределить с помощью директивы SET.

.DEF *имя_р=Регистр*

Директива DEF позволяет присвоить символическое *имя_р* регистру. Регистр может иметь несколько символических имен.

3.5. Директивы задания набора допустимых команд

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

Директива DEVICE сообщает ассемблеру, для какого типа устройства пишется программа.

Если ассемблер встретит команду, которая не поддерживается указанным типом микроконтроллера, то будет выдано сообщение. Сообщение появится и в случае, если размер программы превысит объем имеющейся в этом устройстве памяти.

3.6. Директивы управления файлами

.INCLUDE *"имя файла"*

Директива INCLUDE говорит ассемблеру начать читать из другого файла. Ассемблер будет ассемблировать этот файл до конца файла или до директивы EXIT. Включаемый файл может сам включать директивы INCLUDE.

.EXIT

Директива EXIT позволяет ассемблеру остановить ассемблирование текущего файла.

Обычно ассемблер работает до конца файла. Если он встретит директиву EXIT, то продолжит ассемблировать со строки, следующей за директивой INCLUDE.

3.7. Директивы управления листингом

.NOLIST

Директива выключения генерации lst-файла.

.LIST

Директива включения генерации lst-файла.

4. СИСТЕМА КОМАНД МИКРОКОНТРОЛЛЕРОВ СЕМЕЙСТВА AVR

Набор команд микроконтроллера семейства AVR включает в себя основные группы команд:

– команды пересылки данных;

– арифметические команды;

– логические команды;

– команды перехода.

Команда, написанная на ассемблере, обязательно содержит буквенное обозначение операции (*мнемонику*) и аргументы, которые в ассемблере принято называть *операндами.* Некоторые команды могут не иметь операндов. Ассемблер не различает регистр символов.

При описании команд используются следующие обозначения регистров и операндов:

– **Rd** – регистр назначения (и источник) в регистровом файле;

– **Rr** – регистр-источник в регистровом файле;

– **Rdl** – регистры R24, R26, R28, R30 в инструкциях ADIW и SBIW;

– **K** – литерал, или байт данных (константа 8 бит);

– **b** – номер бита в регистре общего назначения или регистре I/O (константа 3 бита);

– **s** – номер бита в регистре статуса (константа 3 бита);

– **k** – значение адреса (константа или константное выражение);

– **P** – адрес I/O порта (константа 5-6 бит);

– **q** – смещение при прямой адресации (константа 6 бит);

– **X, Y, Z** – регистры косвенной адресации (X = R27:R26, Y = R29:R28, Z = R31:R30);

– **STACK** – стек для адреса возврата и опущенных в стек регистров;

– **SP** – указатель стека.

4.1. Команды пересылки данных

Команды пересылки данных не требуют выполнения никаких операций над операндами. Операнды просто пересылаются (точнее, копируются) из источника (Source) в приемник (Destination). Источником и приемником могут быть внутренние регистры процессора, ячейки памяти или устройства ввода/вывода. Для различных источников и приемников и разных методов адресации используются разные команды, приведенные в табл. 4–7.

Таблица 4

Команды загрузки (записи) содержимого в регистры

Мнемоника	Операнды	Описание	Операция	Флаги
LDI	Rd, K 16 ≤ d ≤ 31 0 ≤ K ≤ 255	Загрузить непосредственное значение	Rd ← K	Нет
LDS	Rd, k 0 ≤ d ≤ 31 0 ≤ k ≤ 65535	Загрузить из ОЗУ	Rd ← (k)	Нет
LD	Rd, X 0 ≤ d ≤ 31	Загрузить косвенно	Rd ← (X)	Нет
LD	Rd, X+ 0 ≤ d ≤ 31	Загрузить косвенно с постинкрементом	Rd ← (X), X ← X + 1	Нет
LD	Rd, X– 0 ≤ d ≤ 31	Загрузить косвенно с преддекрементом	X ← X – 1, Rd ← (X)	Нет
LD	Rd, Y 0 ≤ d ≤ 31	Загрузить косвенно	Rd ← (Y),	Нет
LD	Rd, Y+ 0 ≤ d ≤ 31	Загрузить косвенно с постинкрементом	Rd ← (Y), Y ← Y + 1	Нет
LD	Rd, Y– 0 ≤ d ≤ 31	Загрузить косвенно с преддекрементом	Y ← Y – 1, Rd ← (Y)	Нет
LD	Rd, Z 0 ≤ d ≤ 31	Загрузить косвенно	Rd ← (Z)	Нет
LD	Rd, Z+ 0 ≤ d ≤ 31	Загрузить косвенно с постинкрементом	Rd ← (Z), Z ← Z + 1	Нет
LD	Rd, Z– 0 ≤ d ≤ 31	Загрузить косвенно с преддекрементом	Z ← Z – 1, Rd ← (Z)	Нет
LDD	Rd, Y + q 0 ≤ d ≤ 31 0 ≤ q ≤ 63	Загрузить косвенно со смещением	Rd ← (Y + q)	Нет
LDD	Rd, Z + q 0 ≤ d ≤ 31 0 ≤ q ≤ 31	Загрузить косвенно со смещением	Rd ← (Z + q)	Нет
ELPM	Нет	Расширенная загрузка из памяти программ в регистр R0	R0 ← (Z + RAMPZ)	Нет
LPM	Нет	Загрузить байт из памяти программ	R0 ← (Z)	Нет

Таблица 5

Команды сохранения в памяти содержимого регистров

Мнемоника	Операнды	Описание	Операция	Флаги
STS	k, Rr $0 \le d \le 31$ $0 \le k \le 65535$	Записать непосредственно в ОЗУ	(k) ← Rr	Нет
ST	X, Rr $0 \le r \le 31$	Записать косвенно	(X) ← Rr	Нет
ST	X+, Rr $0 \le r \le 31$	Записать косвенно с постинкрементом	(X) ← Rr, X ← X + 1	Нет
ST	–X, Rr $0 \le r \le 31$	Записать косвенно с преддекрементом	X ← X – 1, (X) ← Rr	Нет
ST	Y, Rr $0 \le r \le 31$	Записать косвенно	(Y) ← Rr	Нет
ST	Y+, Rr $0 \le r \le 31$	Записать косвенно с постинкрементом	(Y) ← Rr, Y ← Y + 1	Нет
ST	–Y, Rr $0 \le r \le 31$	Записать косвенно с преддекрементом	Y ← Y – 1, (Y) ← Rr	Нет
ST	Z, Rr $0 \le r \le 31$	Записать косвенно	(Z) ← Rr	Нет
ST	Z+, Rr $0 \le r \le 31$	Записать косвенно с постинкрементом	(Z) ← Rr, Z ← Z + 1	Нет
ST	–Z, Rr $0 \le r \le 31$	Записать косвенно с преддекрементом	Z ← Z – 1, (Z) ← Rr	Нет
STD	Y + q, Rr $0 \le r \le 31$ $0 \le q \le 63$	Записать косвенно со смещением	(Y + q) ← Rr	Нет
STD	Z + q, Rr $0 \le r \le 31$ $0 \le q \le 63$	Записать косвенно со смещением	(Z + q) ← Rr	Нет

Таблица 6

Команды копирования содержимого из одной области памяти в другую

Мнемоника	Операнды	Описание	Операция	Флаги
PUSH	Rr $0 \le r \le 31$	Сохранить регистр в стеке	STACK ← Rr	Нет
POP	Rd $0 \le r \le 31$	Загрузить данные из стека в регистр	STACK → Rd	Нет
MOV	Rd, Rr $0 \le d \le 31$ $0 \le r \le 31$	Копировать регистр	Rd ← Rr	Нет

Таблица 7

Команды записи в устройства ввода/вывода и чтения из устройств ввода/вывода (обмен информацией производится между регистром и устройством ввода/вывода)

Мнемоника	Операнды	Описание	Операция	Флаги
IN	Rd, P $0 \le d \le 31$ $0 \le P \le 63$	Загрузить данные из порта I/O в регистр	Rd ← P	Нет
OUT	P, Rr $0 \le r \le 31$ $0 \le P \le 63$	Записать данные из регистра в порт I/O	P ← Rr	Нет

4.2. Арифметические команды

Арифметические *команды* рассматривают коды операндов как коды двоичных чисел. Эти команды могут быть разделены на несколько групп.

Команды, приведенные в табл. 8, позволяют выполнять операции сложения и вычитания с операндами различных типов при различных ограничениях.

Таблица 8

Команды операций (сложение, вычитание)

Мнемоника	Операнды	Описание	Операция	Флаги
ADD	Rd, Rr 0 ≤ d ≤ 31 0 ≤ r ≤ 31	Сложить без переноса	Rd ← Rd + Rr	Z, C, N, V, H
ADC	Rd, Rr 0 ≤ d ≤ 31 0 ≤ r ≤ 31	Сложить с переносом	Rd ← Rd + Rr + C	Z, C, N, V, H
ADIW	Rd, K dE{24, 26, 28, 30} 0 ≤ k ≤ 63	Сложить непосредственное значение со словом	Rdh : Rdl ← ← Rdh : Rdl + K	Z, C, N, V
SUB	Rd, Rr 0 ≤ d ≤ 31 0 ≤ r ≤ 31	Вычесть без заема	Rd ← Rd – Rr	Z, C, N, V, H
SUBI	Rd, K 16 ≤ d ≤ 31 0 ≤ k ≤ 255	Вычесть непосредственное значение	Rd ← Rd – K	Z, C, N, V, H
SBC	Rd, Rr 0 ≤ d ≤ 31 0 ≤ r ≤ 31	Вычесть с заемом	Rd ← Rd – Rr – C	Z, C, N, V, H
SBCI	Rd, K 16 ≤ d ≤ 32 0 ≤ k ≤ 255	Вычесть непосредственное значение с заемом	Rd ← Rd–K–C	Z, C, N, V, H
SBIW	Rd, K dE{24, 26, 28, 30} 0 ≤ K ≤ б3	Вычесть непосредственное значение из слова	Rdh : Rdl ← ← Rdh : Rdl – K	Z, C, N, V

Команды инкремента (увеличения на единицу, **INC**) и декремента (уменьшения на единицу, **DEC**) также бывают очень удобны. Их можно в принципе заменить командами суммирования с единицей или вычитания единицы, но инкремент и декремент выполняются быстрее, чем суммирование и вычитание. Эти команды требуют одного входного операнда, который одновременно является и выходным операндом (табл. 9).

Таблица 9

Команды инкремента и декремента

Мнемоника	Операнды	Описание	Операция	Флаги
INC	Rd $0 \le d \le 31$	Инкрементировать	Rd ← Rd + 1	Z, N, V
DEC	Rd $0 \le d \le 31$	Декрементировать	Rd ← Rd – 1	Z, N, V

Команды сравнения, приведенные в табл. 10, предназначены для сравнения двух входных операндов. По сути, они вычисляют разность этих двух операндов, но выходного операнда не формируют, а всего лишь изменяют биты в регистре состояния SREG по результату этого вычитания. Следующая за командой сравнения команда (обычно это *команда перехода*) будет анализировать биты в регистре состояния процессора и выполнять действия в зависимости от их значений.

Таблица 10

Команды сравнения

Мнемоника	Операнды	Описание	Операция	Флаги
CP	Rd, Rr $0 \le d \le 31$ $0 \le r \le 31$	Сравнить	Rd – Rr	Z, C, N, V, H
CPC	Rd, Rr $0 \le d \le 31$ $0 \le r \le 31$	Сравнить с учетом переноса	Rd – Rr – C	Z, C, N, V, H
CPI	Rd, K $16 \le d \le 31$ $0 \le k \le 255$	Сравнить с константой	Rd – K	Z, C, N, V, H

4.3. Логические команды

Логические команды выполняют над операндами логические (побитовые) операции, т.е. они рассматривают коды операндов не как единое число, а как набор отдельных битов. Этим они отличаются от арифметических команд. Команды требуют двух входных операндов и формируют один выходной операнд.

Команды логических операций, сведенные в табл. 11, позволяют побитно вычислять основные логические функции от двух входных операндов.

Кроме того, операция И (AND) используется для принудительной очистки заданных битов (в качестве одного из операндов при этом используется код маски, в котором разряды, требующие очистки, установлены в нуль).

Операция ИЛИ (OR) применяется для принудительной установки заданных битов (в качестве одного из операндов при этом используется код маски, в котором разряды, требующие установки в единицу, равны единице).

Операция «Исключающее ИЛИ» (XOR) используется для инверсии заданных битов (в качестве одного из операндов при этом применяется код маски, в котором биты, подлежащие инверсии, установлены в единицу).

Таблица 11

Команды логических операций

Мнемоника	Операнды	Описание	Операция	Флаги
AND	Rd, Rr $0 \le d \le 31$ $0 \le r \le 31$	Выполнить логическое AND	$Rd \leftarrow Rd \cdot Rr$	Z, N, V
ANDI	Rd, K $16 \le d \le 31$ $0 < k \le 255$	Выполнить логическое AND	$Rd \leftarrow Rd \cdot K$	Z, N, V
OR	Rd, Rr $0 \le d \le 31$ $0 \le r \le 31$	Выполнить логическое OR	$Rd \leftarrow Rd \vee Rr$	Z, N, V
ORI	Rd, K $16 \le d \le 31$ $0 \le k \le 255$	Выполнить логическое OR с непосредственным значением	$Rd \leftarrow Rd \vee K$	Z, N, V
EOR	Rd, Rr $0 \le d \le 31$ $0 \le r \le 31$	Выполнить исключающее OR	$Rd \leftarrow Rd \oplus Rr$	Z, N, V

Команды сдвигов, приведенные в табл. 12, позволяют побитно сдвигать код операнда вправо (в сторону младших разрядов) или влево (в сторону старших разрядов). Тип сдвига (логический, арифметический или циклический) определяет, каково будет новое значение старшего бита (при сдвиге вправо) или младшего бита (при сдвиге влево), а также определяет, будет ли где-то сохранено прежнее значение старшего бита (при сдвиге влево) или младшего бита (при сдвиге вправо).

Для примера на рис. 6 показаны действия, выполняемые командами сдвигов вправо.

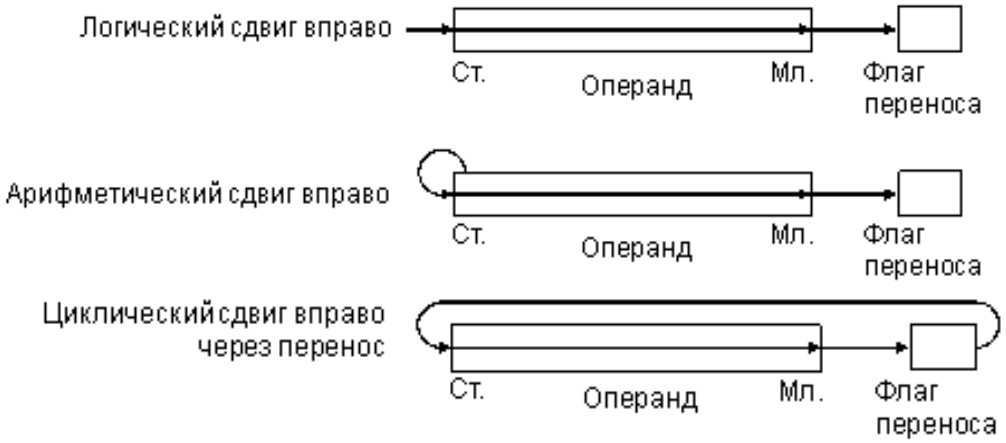


Рис. 6. Команды сдвигов вправо

Таблица 12

Команды сдвигов

Мнемоника	Операнды	Описание	Операция	Флаги
LSL	Rd 0 ≤ d ≤ 31	Логически сдвинуть влево	Rd(n + 1) ← Rd(n), Rd(0) ← 0, C ← Rd(7)	Z, C, N, V, H
LSR	Rd 0 ≤ d ≤ 31	Логически сдвинуть вправо	Rd(n) ← Rd(n + 1), Rd(7) ← 0, C ← Rd(0)	Z, C, N, V
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
ROR	Rd 0 ≤ d ≤ 31	Сдвинуть вправо через перенос	Rd(7) ← C, Rd(n) ← Rd(n + 1), C ← Rd(0)	Z, C, N, V
ASR	Rd 0 ≤ d ≤ 31	Арифметически сдвинуть вправо	Rd(n) ← Rd(n + 1), n = 0...6, C ← Rd(0), Rd(7) = const	Z, C, N, V
SWAP	Rd 0 ≤ d ≤ 31	Поменять нибблы местами	Rd(3...0) <–> Rd(7...4)	Нет

Команды проверки битов и операндов (табл. 13) предназначены для установки или очистки битов регистров в зависимости от значения выбранных битов или всего операнда в целом. Выходного операнда команды не формируют.

Таблица 13

Команды проверки битов и операндов

Мнемоника	Операнды	Описание	Операция	Флаги
TST	Rd $0 \le r \le 31$	Проверить на ноль и минус	Rd ← Rd and Rd	Z, N, V
COM	Rd $0 \le d \le 31$	Выполнить дополнение до единицы (инверсия)	Rd ← $FF – Rd	Z, C, N, V
NEG	Rd $0 \le d \le 31$	Выполнить дополнение до двух	Rd ← $00 – Rd	Z, C, N, V, H
BST	Rd, b $0 \le d \le 31$ $0 \le b \le 7$	Переписать бит из регистра во флаг T	T ← Rd(b)	T
BLD	Rd, b $0 \le d \le 31$ $0 \le b \le 7$	Загрузить T флаг в бит регистра	Rd(b) ← T	Нет
SBI	P, b $0 \le P \le 31$ $0 \le b \le 7$	Установить бит в регистр I/O	I/O (P,b) ← 1	Нет
CBI	P, b $0 \le P \le 31$ $0 \le b \le 7$	Очистить бит в регистре I/O	I/O (P,b) ← 0	Нет
SBR	Rd, K $16 \le d \le 31$ $0 \le k \le 255$	Установить биты в регистре	Rd ← Rd ∨ K	Z, N, V
CBR	Rd, K $16 \le d \le 31$ $0 \le k \le 255$	Очистить биты в регистре	Rd ← Rd · ($FF – K)	Z, N, V
SER	Rd $16 \le d \le 31$	Установить все биты регистра	Rd ← $FF	Нет
CLR	Rd $0 \le d \le 31$	Очистить регистр	Rd ← Rd ⊕ Rd	Z, N, V
NOP	Нет	Выполнить холостую команду	–	Нет
SLEEP	Нет	Установить режим SLEEP	–	Нет
WDR	Нет	Сбросить сторожевой таймер	–	Нет

Команды установки и очистки битов регистра состояния процессора (т.е. флагов) позволяют установить или очистить любой флаг, что бывает очень удобно (табл. 14). Каждому флагу обычно соответствуют две команды, одна из которых устанавливает его в единицу, а другая сбрасывает в нуль.

Таблица 14

Команды установки и очистки битов (флагов)
регистра состояния процессора (SREG)

Мнемоника	Операнды	Описание	Операция	Флаги
BSET	s, $0 \le s \le 7$	Установить флаг	SREG(s) ← 1	SREG(s)
BCLR	s, $0 \le s \le 7$	Очистить флаг	SREG(s) ← 0	SREG(s)
SEI	Нет	Установить флаг глобального прерывания	I ← 1	I
CLI	Нет	Очистить флаг глобального прерывания	I ← 0	I
SET	Нет	Установить флаг T	T ← 1	T
CLT	Нет	Очистить флаг T	T ← 0	T
SEH	Нет	Установить флаг полупереноса	H ← 1	H
CLH	Нет	Очистить флаг полупереноса	H ← 0	H
SES	Нет	Установить флаг знака	S ← 1	S
CLS	Нет	Очистить флаг знака	S ← 0	S
SEV	Нет	Установить флаг переполнения	V ← 1	V
CLV	Нет	Очистить флаг переполнения	V ← 0	V
SEN	Нет	Установить флаг отрицательного значения	M ← 1	N
CLN	Нет	Очистить флаг отрицательного значения	N ← 0	N
SEZ	Нет	Установить флаг нулевого значения	Z ← 1	Z
CLZ	Нет	Очистить флаг нулевого значения	Z ← 0	Z
SEC	Нет	Установить флаг переноса	C ← 1	C
CLC	Нет	Очистить флаг переноса	C ← 0	C

4.4. Команды перехода

Команды переходов предназначены для организации всевозможных циклов, ветвлений, вызовов подпрограмм и т.д., т.е. они нарушают последовательный ход выполнения программы.

Команды безусловных переходов, приведенные в табл. 15, вызывают переход в новый адрес независимо ни от чего. Они могут вызывать переход на указанную величину смещения (вперед или назад) или же на указанный адрес памяти. Величина смещения или новое значение адреса указываются в качестве входного операнда.

Таблица 15

Команды безусловных переходов

Мнемоника	Операнды	Описание	Операция	Флаги
RJMP	k –2K < k < 2K	Перейти относительно	PC ← PC + k + 1	Нет
JMP	k 0 < k < 4M	Перейти	PC ← k	Нет
LJMP	Нет	Перейти косвенно	PC ← Z	Нет

Некоторые *команды переходов* предусматривают в дальнейшем возврат назад, в точку, из которой был сделан переход (табл. 16). Если возврат предусмотрен, то текущие параметры процессора сохраняются в стеке.

Таблица 16

Команды безусловных переходов с возвратом

Мнемоника	Операнды	Описание	Операция	Флаги
RCALL	k –2K ≤ k ≤ 2K	Вызвать подпрограмму относительно	PC ← PC + k + 1	Нет
CALL	k 0 ≤ k ≤ 64K	Выполнить длинный вызов подпрограммы	PC ← k	Нет
ICALL	Нет	Вызвать подпрограмму косвенно	PC ← Z	Нет
RET	Нет	Вернуться из подпрограммы	PC ← STACK	Нет
RETI	Нет	Вернуться из прерывания	PC ← STACK	I

Команды условных переходов вызывают переход не всегда, а только при выполнении заданных условий (табл. 17).

Таблица 17

Команды условных переходов

Мнемоника	Операнды	Описание	Операция	Флаги
CPSE	Rd, Rr 0≤ d ≤31 0 ≤ r ≤ 31	Сравнить и пропустить, если равно	if Rd = Rr then PC ← PC + 2 (or 3)	Нет
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
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
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
SBIS	P, b 0 ≤ r ≤ 31 0 ≤ b ≤ 7	Пропустить, если бит в регистре I/O установлен	if l/O P(b) = 1 then PC ← PC + 2 (or 3)	Нет

Часто в качестве условий выступают значения флагов в регистре состояния процессора (SREG). То есть условием перехода является результат предыдущей операции, меняющей значения флагов (табл. 18).

Таблица 18

Команды условных переходов по состоянию флагов в регистре SREG

Мнемоника	Операнды	Описание	Операция	Флаги
1	2	3	4	5
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
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
BREQ	k –64 ≤ k ≤ +63	Перейти, если равно	if Rd = Rr (Z = 1) then PC ← PC + k + 1	Нет
BRNE	k –64 ≤ k ≤ +63	Перейти, если не равно	if Rd ≠ Rr (Z = 0) then PC ← PC + k + 1	Нет

Окончание табл. 18

1	2	3	4	5
BRSH	k –64 ≤ k ≤ +63	Перейти, если равно или больше (без знака)	if Rd < Rr (C = 0) then PC ← PC + k+ 1	Нет
BRLO	k –64 ≤ k ≤ +63	Перейти, если меньше (без знака)	if Rd < Rr (C = 1) then PC ← PC + k + 1	Нет
BRGE	k –64 ≤ k ≤ +63	Перейти, если больше или равно (с учетом знака)	if Rd > Rr (N ⊕ V = 0) then PC ← PC + k + 1	Нет
BRLT	k –64 ≤ k ≤ +63	Перейти, если меньше чем (со знаком)	if Rd < Rr (N ⊕ V = 1) then PC ← PC + k+ 1	Нет
BRMI	k –64 ≤ k ≤ +63	Перейти, если минус	if N = 1 then PC ← PC + k + 1	Нет
BRPL	k –64 ≤ k ≤ +63	Перейти, если плюс	if N = 0 then PC ← PC + k + 1	Нет
BRHS	k –64 ≤ k ≤ +63	Перейти, если флаг полупереноса установлен	if H = 1 then PC ← PC + k + 1	Нет
BRHC	k –64 ≤ k ≤ +63	Перейти, если флаг полупереноса очищен	if H = 0 then PC ← PC + k + 1	Нет
BRTS	k –64 ≤ k ≤ +63	Перейти, если флаг T установлен	if T = 1 then PC ← PC + k + 1	Нет
BRTC	k –64 ≤ k ≤ +63	Перейти, если флаг T очищен	if T = 0 then PC ← pc + k + 1	Нет
BRVS	k –64 ≤ k ≤ +63	Перейти, если флаг переполнения установлен	if V = 1 then PC ← PC + k + 1	Нет
BRVC	k –64 ≤ k ≤ +63	Перейти, если флаг переполнения очищен	if V = 0 then PC ← PC + k + 1	Нет
BRIE	k –64 ≤ k ≤ +63	Перейти, если глобальное прерывание разрешено	if I = 1 then PC ← PC + k + 1	Нет
BRID	k –64k ≤ k ≤ +63	Перейти, если глобальное прерывание запрещено	if I = 0 then PC ← PC + k + 1	Нет

5. ПРОГРАММА НА АССЕМБЛЕРЕ

5.1. Структура программы

Программа, написанная на ассемблере, должна иметь определенную структуру. Ниже приводится рекомендованный для AT90S8535 шаблон. Курсив для выделения комментариев используется в этом примере для наглядности. В реальной программе ассемблер не поддерживает различное форматирование шрифтов.

```
;*****************************************************
; название программы,
; краткое описание, необходимые пояснения
;*****************************************************
;******подключаемые дополнительные файлы***********
.include “8535def.inc”                    ; файл описания AT90S8535
.include «имя_файла1.расширение»          ; включение дополнительных
.include «имя_файла2.расширение»          ; файлов
;******глобальные константы***********
.equ          имя1 = xxxx ;
.equ          имя2 = nnnn
;******глобальные регистровые переменные***********
.def          имя1= регистр
.def          имя2= регистр
;*******сегмент данных***********
.dseg
.org xxxx     ; 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    xxxx                        ; 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 IRQ0
.org    $0004
rjmp INT1     ; обработчик прерывания IRQ1
.org    adrINTx       ; адрес следующего обработчика прерываний
rjmp  INTx    ; обработчик прерывания x
; далее по порядку располагать обработчики остальных прерываний

;*******начало основной программы**********
main:  <команда> xxxx
          …       …

;******* подпрограммы  *********
;*******подпрограмма 1**********
subr1: <команда> xxxx
……      …….          ……
          ret
;*******подпрограмма 2**********
subr2: <команда> xxxx
……      …….          ……
          ret
…….
;******* программы обработчиков прерываний**********
INT0: <команда> xxxx
……    …….          ……
          reti
INT1: <команда> xxxx
……    …….          ……
          reti
INTx: <команда> xxxx
……    …….          ……
          reti
……    ……          ……
; конец программы никак не обозначается.
```

5.2. Алгоритм решения задачи и структура данных

Алгоритм программы состоит из двух частей: последовательности операторов, формирующих структуру данных, и набора операторов, реализующих метод решения задачи. Составление алгоритма решения задачи и проектирование структуры данных рассматривается на простейшем примере.

Пример. Составьте программу вычитания из числа 5 числа 3. Если на блоке управления включен тумблер SA1, то на индикацию должен выдаваться результат вычитания. Если тумблер SA1 отключен, на индикацию выводится цифра ноль.

Алгоритм решения задачи может быть следующим:

– проверяется состояние ключа SA1;

– если ключ SA1 включен, то необходимо произвести вычитание и запомнить результат и перейти к индикации;

– если ключ SA1 выключен, то необходимо просто запомнить ноль и перейти к индикации;

– задать или сформировать адрес ячейки, в которой хранится код для индикации результата;

– выдать результат на индикацию;

– вернуться к проверке состояния ключа SA1 (если предполагается бесконечное выполнение программы).

Параллельно с алгоритмом решения задачи необходимо проектировать и структуру данных – порядок настройки портов, перечень входных и выходных переменных и их адреса, вершину стека. Структуру данных можно представить в виде совокупности нескольких таблиц (табл. 19–22).

Таблица 19

Порядок настройки портов

Порты	Выводы на вход	Выводы на выход
Порт A	0–7	–
Порт B	0, 1, 4–7	2, 3
Порт C	–	0–7
Порт D	0–3	4–7

Таблица 20

Размещение структурных элементов программы

Элемент структуры данных	Имя	Адрес	Память
Вершина стека	SP	$025f	Регистры I/O
Начало программы	–	$0030	FLASH

Таблица 21

Перечень констант

Константы	Имя	Значение	Адрес	Память
Адрес кода 0	cod0	$3f	$64	SRAM
Адрес кода 1	cod1	$06	$65	SRAM
Адрес кода 2	cod2	$5b	$66	SRAM
Адрес кода 3	cod3	$4f	$67	SRAM
Адрес кода 4	cod4	$66	$68	SRAM
Адрес кода 5	cod5	$6d	$69	SRAM
Адрес кода 6	cod6	$7d	$6a	SRAM
Адрес кода 7	cod7	$07	$6b	SRAM
Адрес кода 8	cod8	$7f	$6c	SRAM
Адрес кода 9	cod9	$6f	$6d	SRAM

Таблица 22

Перечень переменных

Переменные	Имя	Значение	Адрес	Память
Уменьшаемое	–	5	r17	РОН
Вычитаемое	–	3	r19	РОН
Результат (разность)	–	–	r20	РОН
Адрес ячейки кода индикации разности	–	–	r0	РОН
Адрес первого из кодов индикации	–	$64	Z (zh, zl) (r31, r30)	РОН
Вспомогательные регистры для промежуточных значений	–	–	r16, r17	РОН

5.3. Пример алгоритма программы

Алгоритм программы решения задачи представлен на рис. 7. В начале программы размещаются операторы, организующие требуемую структуру данных, а затем операторы решения задачи.

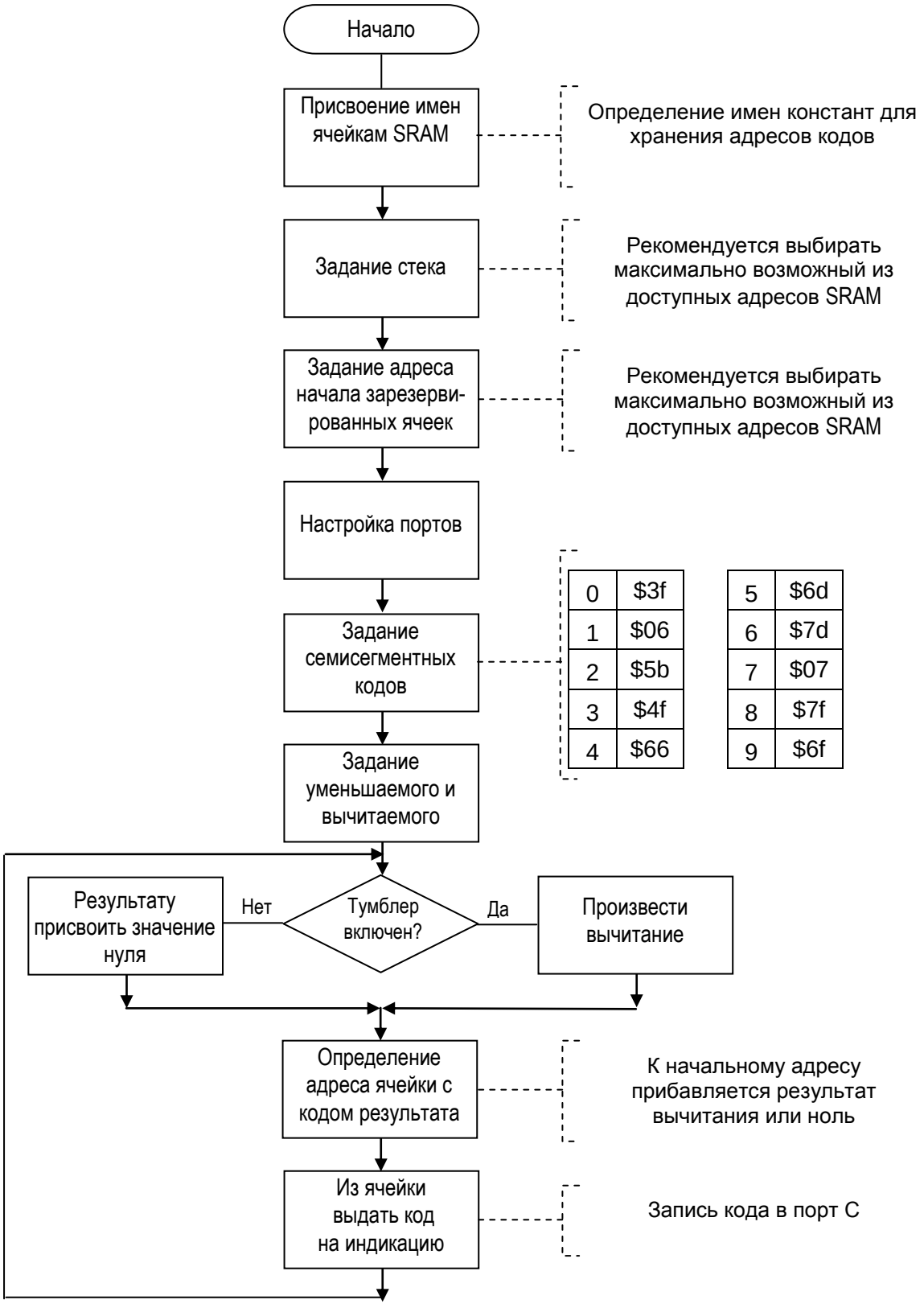


Рис. 7. Алгоритм программы

5.4. Пример текста программы

Полученный алгоритм реализуется программой № 1. Один блок алгоритма может реализовываться несколькими командами.

```
;***************************************************
; Программа № 1. Использование директивы EQU
;************************************ **********************
;******подключаемые дополнительные файлы******
.include "8535def.inc"        ; включить файл – описание для AT90S8535

.dseg                         ; объявление сегмента данных

;******* ; присвоение имен ячейкам SRAM **********************
.equ   cod0=$64
.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                         ; объявление сегмента кодов
.org   0                      ; адрес начала программы в программной памяти
rjmp   reset                  ; прерывание по сбросу

;*******начало основной программы**********
.org   $30
reset:
; ******* задание стека с вершиной по адресу $025f *******
ldi    r16,$02
out    sph,r16
ldi    r16,$5f
out    spl,r16

; ******* задание адреса начала зарезервированных ячеек  *******
```

```
ldi     zl,$64
ldi     zh,$00

; *******    Настройка портов     ****************************
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 (активен HG2)

; *******   задание семисегментных кодов    *******
ldi     r17,$3f
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

;******* Задание уменьшаемого и вычитаемого    *******
ldi     r17,5                   ; задание уменьшаемого
```

```
ldi     r18,3                   ; задание вычитаемого

m1:

;******* Проверка «Тумблер включен?»    *******
sbis pina,4             ; если включен тумблер SA1, то пропустить
rjmp    m2              ; следующую команду

;******* Произвести вычитание     *******
mov     r20,r17         ; в r20 поместить уменьшаемое
sub     r20,r18         ; вычесть вычитаемое, результат остается в r20
rjmp    vv

;******* Результату присвоить значение нуля     *******
m2:
ldi     r20,0           ; в r20 записать ноль

;******* Определение  адреса ячейки с кодом результата     *******
vv:
push    zl                      ; сохранить zl в стеке
add     zl,r20                  ; сложить zl с результатом
ld      r0,z                    ; семисегментный код результата переслать в r0
pop     zl                      ; извлечь zl из стека

;******* Из ячейки  выдать код  на индикацию     *******
out     portc,r0                ; запись кода в порт C

rjmp    m1
```

6. НАБОР И ОТЛАДКА ПРОГРАММ

6.1. Набор программы

Для ввода программы используется программа Avr Studio. Для запуска программы запустите файл AvrStudio.exe. Появится основное диалоговое окно программы (рис. 8).

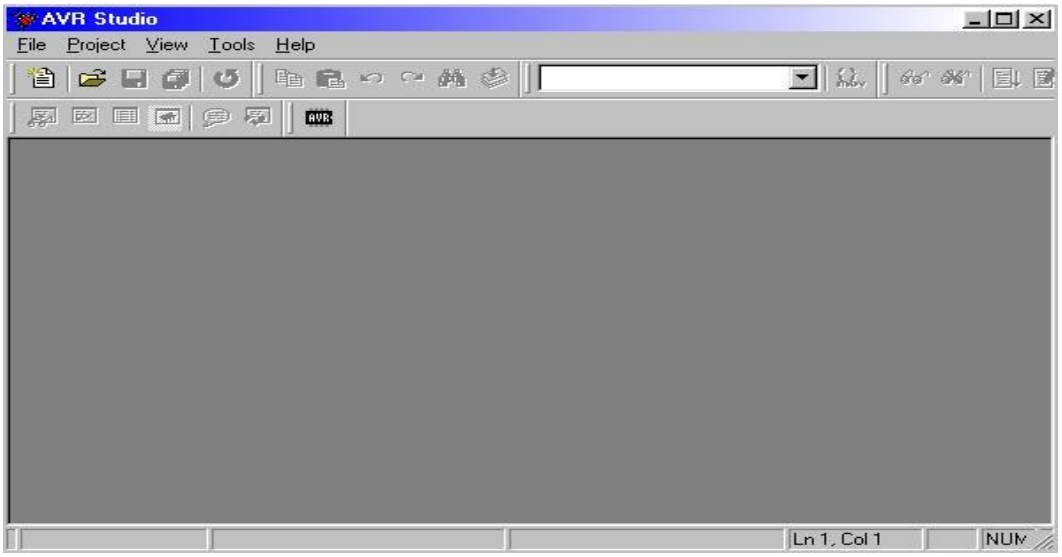


Рис. 8. Диалоговое окно программы Avr Studio

В верхней части программы находится меню, в нем надо выбрать **Project–>New.**

В появившемся окне выберите имя проекта (Project name), место на диске, куда сохранять проект (Location), а также тип проекта (Project type), щелкнув мышью на **AVR assembler**, затем щелкнув на кнопке OK (рис. 9).

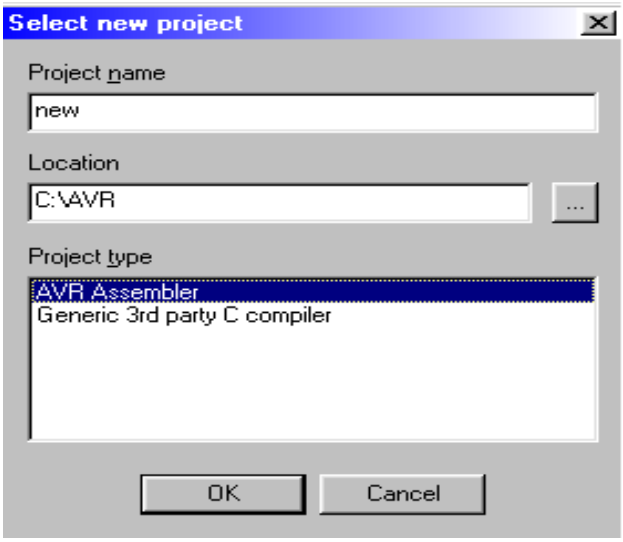


Рис. 9. Добавление проекта (шаг 1)

Появится окно проектов (Project :), в котором находится дерево файлов. В него входят файлы, которые будут компилироваться, с расширением .asm (Assembler Files), а также дополнительные файлы библиотек (Other Files).

На ветви **Assembler Files** необходимо щелкнуть правой кнопкой мыши. Появится меню, в нем необходимо выбрать пункт **Create New File** (рис. 10).

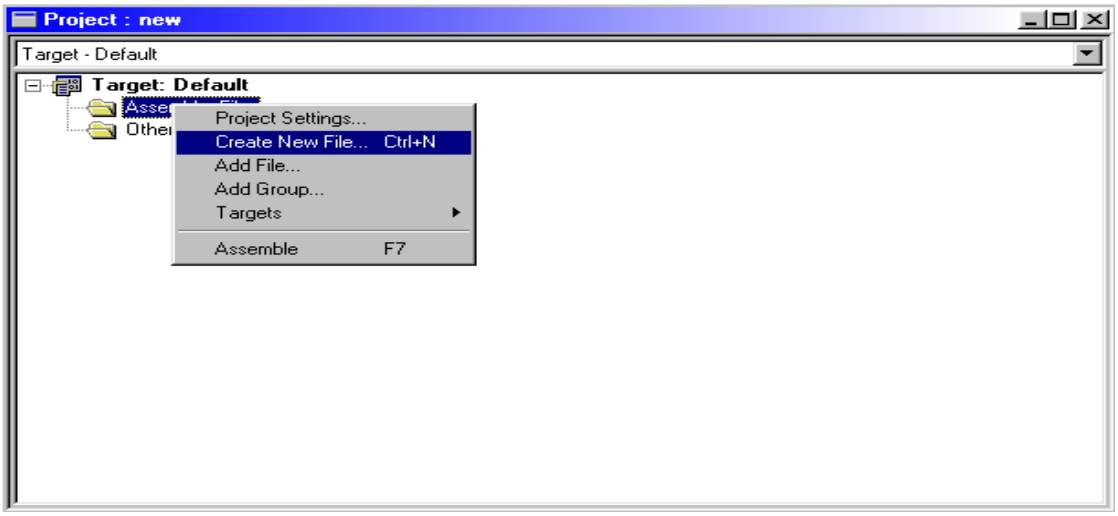


Рис. 10. Добавление проекта (шаг 2)

В появившемся окне выберите имя файла (Name), **обязательно с расширением .asm,** и нажмите кнопку OK (рис. 11).

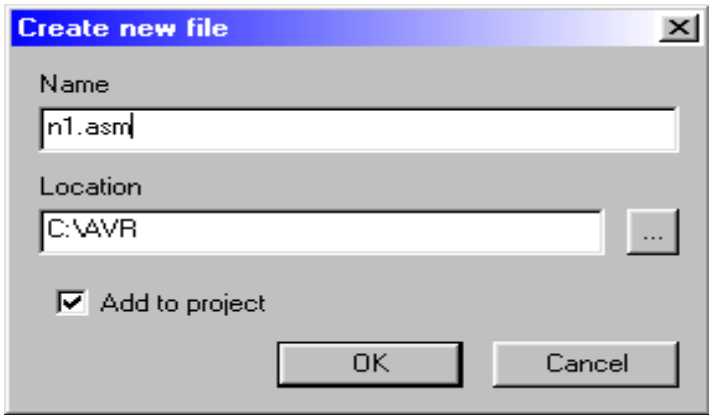


Рис. 11. Добавление проекта (шаг 3)

В открывшемся окне проектов имя файла появится в дереве файлов отдельной ветвью. Его нужно перетащить, удерживая на нем правую кнопку мыши, на ветвь Assembler Files. После этого, щелкнув правой клавишей мыши, снова вызовите меню и выберите пункт **Параметры проекта** (**Project Settings**). В появившемся окне AVR **Assembler Options** в пункте **Формат выходного файла (Output file format:)** выберите **Intel Intellec 8/MDS (Intel Hex)** и нажмите OK (рис. 12).

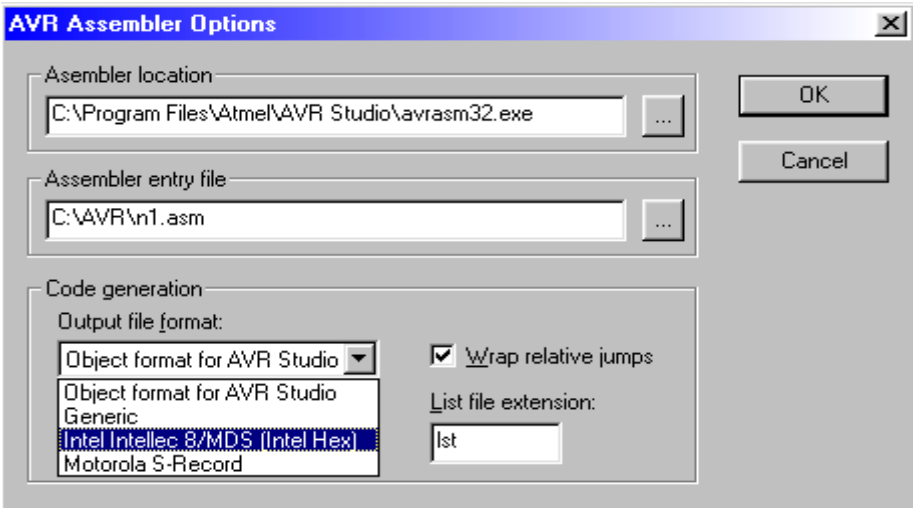


Рис. 12. Добавление проекта (шаг 4)

Затем снова вернитесь в окно проектов и, щелкнув правой кнопкой мыши на ветви **Другие файлы (Other Files),** вызовите меню и выберите пункт **Добавить файл (Add File)**, найдите файл **8535def.inc** и подключите его к проекту.

Необходимо отметить, что этот файл должен находиться в той же папке, что и asm-файл, который создается, поэтому его лучше скопировать заранее из папки **Programm files \ Atmel \ AVR Studio \ Appnotes.**, иначе это вызовет ошибку компиляции. Если все сделано правильно, то окно проектов должно выглядеть следующим образом (рис. 13).

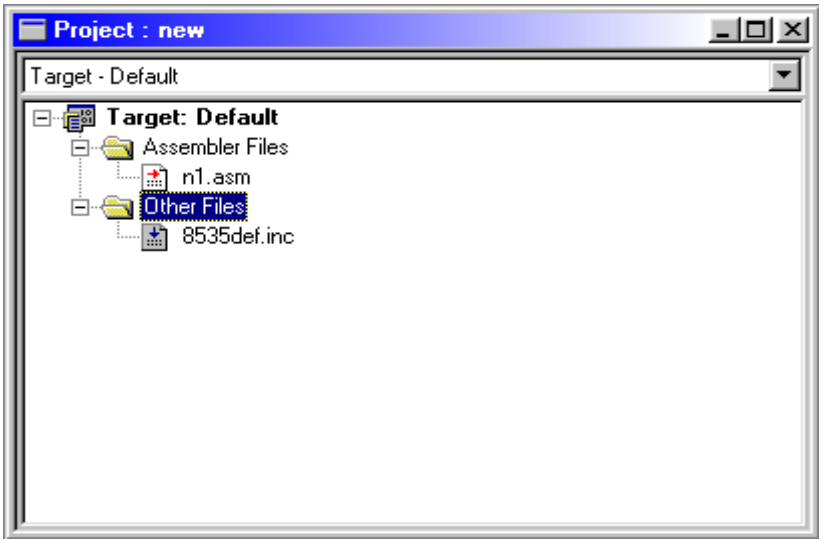


Рис. 13. Добавление проекта (шаг 5)

Теперь щелкните два раза на asm-файле и в открывшемся окне наберите программу.

После того как программа набрана, нажмите **F7** и произведите ее компиляцию, при этом создается файл с расширением **hex**, который затем надо будет записать в микроконтроллер.

После компиляции появится окно **Project Output,** в котором указано, какой файл ассемблируется, используемый файл библиотеки, количество слов в программе и сообщение об отсутствии ошибок: **Assembly complete with no errors.** Если есть ошибки, то в этом окне указывается тип ошибки, номер строки с ошибкой и в конце – общее число ошибок. Для их исправления необходимо вернуться к редактируемому файлу и их исправить, а затем снова откомпилировать программу.

Avr Studio позволяет не только компилировать программы, но и отлаживать их на этапе разработки. При этом Avr Studio эмулирует работу микроконтроллера, всех портов ввода/вывода, счетчиков/таймеров, прерываний, ШИМ и АЦП. Эмуляция работы программы позволяет рассмотреть ее работу, как если бы она была записана в микроконтроллер.

Необходимо отметить, что можно эмулировать работу только программ, не содержащих ошибок. Поэтому перед эмуляцией Avr Studio произведет компиляцию программы, и если есть ошибки, то эмулировать (отладить) программу не удастся.

6.2. Отладка программы

Для отладки программы в папку с **asm-файлом** необходимо скопировать файл **AT90S8535_Std.aio,** который находится в папке **Programm files \ Atmel \ AVR Studio \ IOSetup.**

В меню **Project** выберите пункт **Build and run** или нажмите **Ctrl + F7**. Появится окно **Опции эмулятора** (**Simulation Options**) – рис. 14.

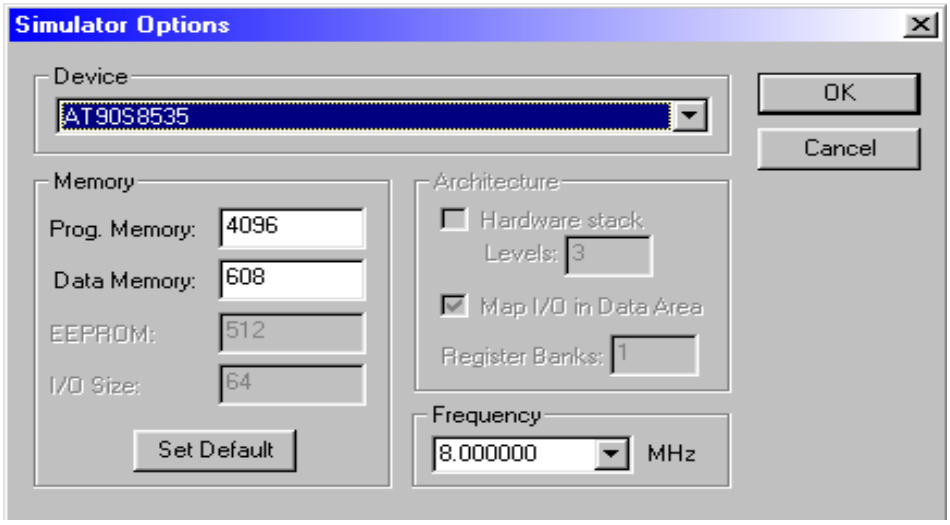


Рис. 14. Окно «Опции эмулятора»

В пункте **Устройство (Device)** нужно выбрать микроконтроллер **AT90S8535**, в пункте **Частота (Frequency)** – частоту **8 МГц** и нажать кнопку OK.

После этого появится окно, в котором набиралась программа, но начало программы будет отмечено желтой стрелкой, выше идут директивы компилятора. При эмуляции работы программы необходимо видеть состояние регистров, портов ввода/вывода, процессора. В главном меню программы выберите пункт **Просмотр** (**View**), затем пункт **Регистры** (**Registers**), далее пункты **Процессор** (**Processor**), **Просмотр ввода/вывода** (**New IO View**).

В меню **View** имеются и другие пункты, которые можно использовать, но в данном пособии они не рассматриваются. Для наблюдения работы микроконтроллера в большинстве случаев достаточно только этих окон. Таким образом, после всех этих действий получится окно примерно такого вида (рис. 15).

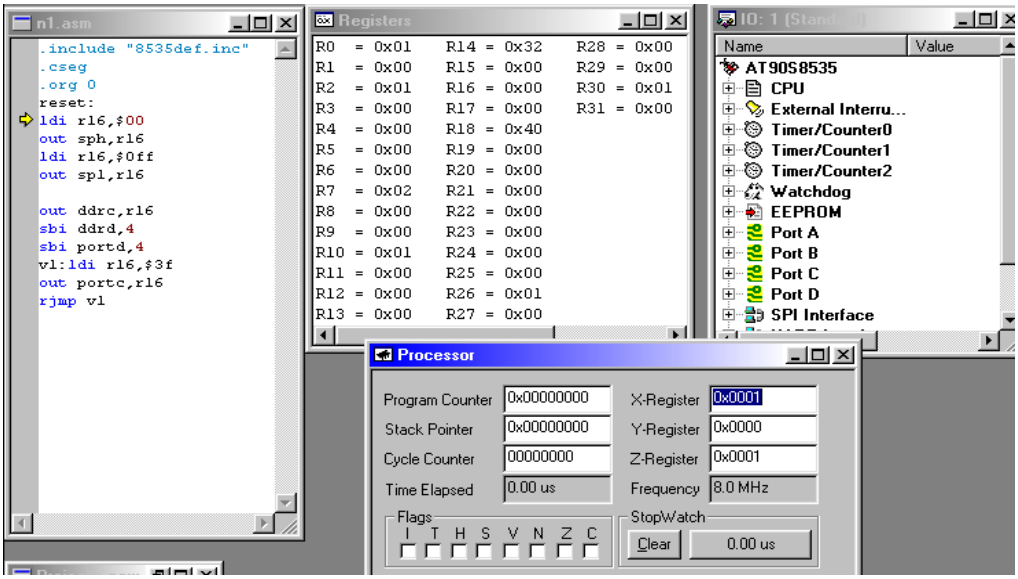


Рис. 15. Наблюдение работы микроконтроллера

Теперь можно приступить к запуску программы. Avr Studio позволяет запустить программу в реальном времени, в пошаговом режиме, до указателя.

В главном меню, в пункте **Отладка** (**Debug**) находятся все варианты запуска программы:

– **Reset** – сброс на начало программы (желтая стрелка указателя показывает на начало);

– **Go** – запуск в реальном времени (программа будет выполняться до тех пор, пока не будет выбран пункт **Break**);

– **Step over** – пошаговый режим (программа выполняется построчно, при этом останавливается после каждой команды, стрелка указывает на текущую команду);

– **Run to cursor** – выполнять до курсора (программа выполняется до места, отмеченного курсором в окне с редактируемой программой).

Во время выполнения программы можно наблюдать за состоянием регистров после каждой команды, тем самым проверяется правильность операций, производимых микроконтроллером.

Наиболее удобный режим для этого – пошаговый.

Содержание окон для наблюдения процессов в микроконтроллере в основном понятно, необходимо пояснить содержание окна **IO**, в котором показаны все устройства микроконтроллера. Напротив каждого устройства стоит знак «+», щелкнув на нем мышкой, получаем содержимое этого устройства, т.е. состояние управляющих регистров, регистров данных и т.д.

Два раза щелкнув на содержании какого-нибудь регистра, можно изменить его состояние в процессе выполнения программы.

В регистре портов ввода/вывода можно задать входные сигналы, отмечая галочкой в нужном бите состояние логической единицы, тем самым эмулируется воздействие внешних сигналов.

В данном пособии не преследуется цель описать все возможности программы Avr Studio, остальное изучается пользователем в процессе работы с программой.

6.3. Запись программы в микроконтроллер

Avr Studio позволяет записывать программу в микроконтроллер, но в стенде используется несколько другая схема программатора, поэтому использовать эту возможность программы нельзя.

Для записи программы в микроконтроллер используется программа **New_SP**. Для запуска программы запустите **new_sp.exe** (рис. 16).

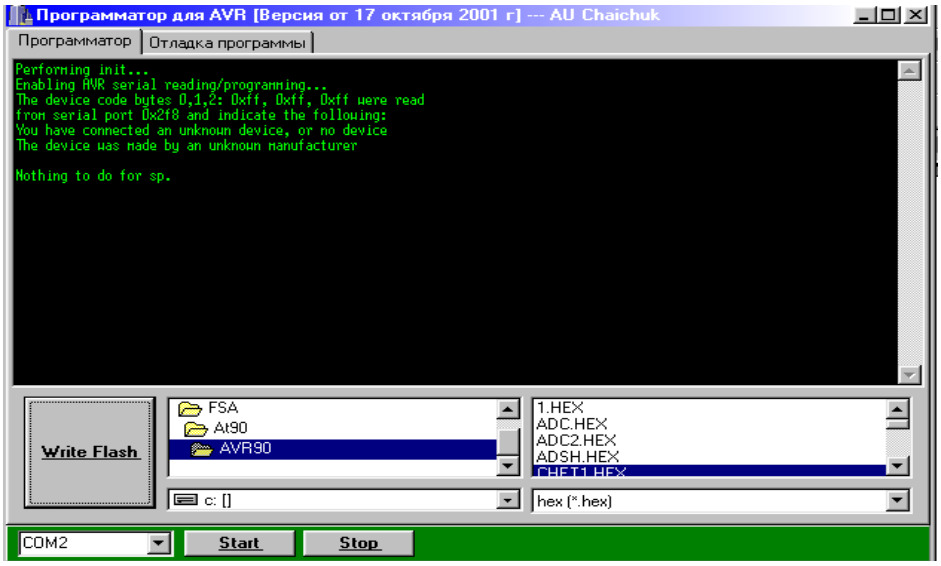


Рис. 16. Запись программы в микроконтроллер

В нижней части окна программы установите номер порта COM2, нажмите кнопку **Start**. Если порт установлен, нижняя часть окна станет зеленой. Затем в окнах с полосами прокрутки выберите диск, каталог и имя записываемого файла с расширением **.hex.** Для записи программы нажмите кнопку **Write Flash**. Программа записывается в микроконтроллер и в верхнем окне выводится тип микроконтроллера и имя записываемого файла. Если запись невозможна, не выбран **hex** файл или нет соединения с микроконтроллером, программа выводит сообщение **Nothing to do for sp.**

ЧАСТЬ II. ЛАБОРАТОРНЫЕ РАБОТЫ

Общие методические указания к лабораторным работам

К каждой лабораторной работе в сжатой форме даны пояснения, позволяющие при знании общих принципов работы изучаемых узлов микроконтроллера подготовиться к лабораторным работам без привлечения дополнительной литературы.

В разделах 3, 4 и 5 представлены директивы ассемблера, развернутая система команд микроконтроллеров семейства AVR, структура и пример программы.

Все лабораторные работы по методике и последовательности выполнения однотипны и заключаются в выполнении индивидуального задания по изучаемой системе (узлу) программируемого микроконтроллера. Поэтому содержание работ и требования к отчетам по выполненной работе тоже однотипны. Ниже приводятся типовой порядок выполнения работ и требования к отчетам по лабораторным работам.

При защите лабораторной работы студент должен быть готов ответить на контрольные вопросы, приведенные в конце каждой лабораторной работы.

Порядок выполнения лабораторных работ

1. Ознакомьтесь с описанием лабораторной работы и индивидуальным заданием, изучите теоретический материал.
2. Дома при подготовке к работе составьте схему алгоритма решения задачи индивидуального задания и структуру данных.
3. Напишите программу на языке ассемблера в любом текстовом редакторе.
4. В компьютерном классе с помощью программы AVR Studio создайте новый проект, введите программу и откомпилируйте ее.
5. При отсутствии синтаксических ошибок в программе проверьте ее выполнение в пошаговом режиме с помощью симулятора AVR Studio.
6. При отсутствии семантических ошибок в программе запишите ее в лаборатории через программатор в ПЗУ микроконтроллера.
7. На рабочем месте включите программируемый микроконтроллер для выполнения заданной задачи. Путем подачи на входы микроконтроллера необходимых входных сигналов и визуального наблюдения выходных сигналов, выдаваемых микроконтроллером, оцените правильность работы подготовленной программы.
8. Выведите на печать (на принтер) отлаженную программу или ее листинг.

Содержание отчета

Отчет по лабораторной работе должен содержать:

а) цель работы;

б) условие индивидуального задания;

в) структуру данных;

г) граф-схему алгоритма решения поставленной задачи и ее краткое описание;

д) программу на языке ассемблера или листинг программы;

е) результаты проверки правильности функционирования программы (в какой последовательности подавались входные сигналы, что визуально наблюдалось при этом и т.п.);

ж) выводы по работе.

Пример оформления отчета по лабораторной работе представлен в Приложении.

Меры безопасности при эксплуатации комплекса

При эксплуатации лабораторного комплекса соблюдайте следующие меры безопасности:

а) питание комплекса осуществляется от удлинителя-размножителя с розетками, имеющими заземляющие выводы; подключение питания составляющих стенда к удлинителю-размножителю осуществляется только через шнуры с вилками, имеющими заземляющие выводы;

б) переключение **LPT** и **COM2** портов системного блока ПЭВМ осуществляется только при отключенной из розетки вилке питания системного блока ПЭВМ (или при обесточенной розетке);

в) все переключения кабелей рабочих мест на блоке связи с ЭВМ осуществляются только при отключенном тумблере блока питания лабораторного комплекса.

Наиболее уязвимыми узлами комплекса являются **LPT** и **COM2** порты ПЭВМ. Соблюдение указанных выше мер обеспечит нормальную работу портов ПЭВМ и комплекса в целом.

КУРСОВОЙ ПРОЕКТ

ЧАСТЬ 1. ИЗУЧЕНИЕ СИСТЕМЫ КОМАНД МИКРОКОНТРОЛЛЕРА И ДИРЕКТИВ АССЕМБЛЕРА

Цель работы

Ознакомление с лабораторным комплексом «Микроконтроллеры и автоматизация», получение навыков работы с программой AVR Studio, ознакомление с системой команд программируемого микроконтроллера AT90S8535, подготовка простейшей программы, ее отладка, запись в микроконтроллер и демонстрация работы подготовленной программы.

Пояснения к работе

Так как рассматриваемая работа первая, то для приобретения обучающимися навыков работы с лабораторным комплексом все обучающиеся сначала делают одинаковую работу.

Со своих рабочих мест они вводят в ПЭВМ одну и ту же программу, приведенную в разделе 5. После компиляции программы ее работа демонстрируется преподавателю.

После такого знакомства с комплексом обучающийся приступает к выполнению индивидуального задания. Он должен найти отличия в структуре данных и алгоритме решения задачи и внести соответствующие изменения в программу. После компиляции программа записывается в микроконтроллер рабочего места и ее работа демонстрируется преподавателю.

Система команд микроконтроллеров семейства AVR представлена в разделе 4.

При использовании команд IN и OUT используются адреса ввода/вывода с $00 по $3F. Но к ним же можно обращаться и как к ячейкам внутреннего ОЗУ. При этом к непосредственному адресу регистра ввода/вывода необходимо прибавить $20.

Следует помнить, что регистры ввода/вывода в пределах адресов от $00 по $1F имеют программно доступные биты. Обращение к ним осуществляется командами SBI и CBI, а проверка состояния – командами SBIS и SBIC.

При арифметических операциях используется регистр состояния SREG, располагаемый по адресу $3F($5F). Формат этого регистра следующий (рис. 17).

Бит	7	6	5	4	3	2	1	0
Имя бита	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
Чтение/Запись	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Исходное значение	0	0	0	0	0	0	0	0

Рис. 17. Регистр состояния – **SREG.** Адрес – $3F($5F)

Бит 7 (I) – разрешение всех прерываний. Для разрешения прерываний этот бит должен быть установлен в состояние 1. Управление разрешением конкретного прерывания выполняется регистром маски прерывания EIMSK и TIMSK. Если этот бит очищен (= 0), то ни одно из прерываний не обрабатывается. Бит аппаратно очищается после возникновения прерывания и устанавливается для последующего разрешения прерывания командой RETI.

Бит 6 (T) – бит сохранения копии. Команды копирования бита **BLD** и **BST** используют этот бит как источник и приемник при операциях с битами. Командой **BST** бит регистра общего назначения копируется в бит **T**, командой BLD бит **T** копируется в бит регистра общего назначения.

Бит 5 (H) – флаг полупереноса. Он указывает на перенос между тетрадами при выполнении ряда арифметических операций.

Бит 4 (S) – бит знака. Бит S имеет значение результата операции, исключающего ИЛИ (N⊕V) над флагами отрицательного значения (N) и дополнения до двух флага переполнения (V).

Бит 3 (V) – дополнение до двух флага переполнения. Он поддерживает арифметику дополнения до двух.

Бит 2 (N) – флаг отрицательного значения. Этот флаг указывает на отрицательный результат ряда арифметических и логических операций.

Бит 1 (Z) – флаг нулевого значения. Этот флаг указывает на нулевой результат ряда арифметических и логических операций.

Бит 0 (C) – флаг переноса. Этот флаг указывает на перенос при арифметических и логических операциях.

Микроконтроллер AT90S8535 имеет четыре параллельных порта ввода/вывода **A, B, C** и **D**. Все четыре параллельных порта являются восьмиразрядными двунаправленными портами.

Взаимодействие с каждым из портов осуществляется через три регистра в пространстве ввода/вывода памяти данных:

– регистр данных;

– регистр направления данных;

– регистр входных данных.

Через регистры данных осуществляется обмен данными с внешними устройствами.

Значения битов в регистре направления данных определяют, как настроены соответствующие выводы порта. Когда бит сброшен (равен нулю), то вывод порта настроен на прием информации от внешнего источника. А если бит установлен (равен единице), то вывод настроен на выдачу информации внешнему приемнику.

Регистры данных и направления данных обеспечивают возможность чтения и записи.

Регистры входных данных обеспечивают только возможность чтения. Обращение к регистрам входных данных обеспечивает чтение физического состояния каждого вывода порта.

Каждый из портов имеет свои особенности использования. Например, порт **A** служит также для ввода аналоговых сигналов **A/D**. Выводы портов **B, C, D** могут выполнять альтернативные функции.

Ниже приводятся обозначения портов, их выводов, альтернативных функций выводов и элементов блока управления, подключенных к этим выводам (табл. 23–26, рис. 18–25).

Таблица 23

Регистр данных порта A – **PORTA.** Адрес – $1B($3B)

Бит	Имя бита	Подключенный элемент блока управления	Вывод порта	Альтернативная функция
7	**PORTA7**	–	**PA7**	Нет
6	**PORTA6**	**SA3**	**PA6**	Нет
5	**PORTA5**	**SA2**	**PA5**	Нет
4	**PORTA4**	**SA1**	**PA4**	Нет
3	**PORTA3**	**SB3**	**PA3**	Нет
2	**PORTA2**	**SB2**	**PA2**	Нет
1	**PORTA1**	**SB1**	**PA1**	Нет
0	**PORTA0**	–	**PA0**	Нет

Бит	7	6	5	4	3	2	1	0
Имя бита	**DDA7**	**DDA6**	**DDA5**	**DDA4**	**DDA3**	**DDA2**	**DDA1**	**DDA0**

Рис. 18. Регистр направления данных порта A – **DDRA**. Адрес – $1A($3A)

Бит	7	6	5	4	3	2	1	0
Имя бита	**PINA7**	**PINA6**	**PINA5**	**PINA4**	**PINA3**	**PINA2**	**PINA1**	**PINA0**

Рис. 19. Регистр входных данных порта A – **PINA.** Адрес – $19($39)

Таблица 24

Регистр данных порта B – **PORTB.** Адрес – $18($38)

Бит	Имя бита	Подключенный элемент блока управления	Вывод порта	Альтернативная функция
7	**PORTB7**	–	**PB7**	SCK – тактовый сигнал SPI
6	**PORTB6**	–	**PB6**	MISO – установка ведущий вход/ведомый выход SPI
5	**PORTB5**	–	**PB5**	MOSI – установка ведущий выход/ведомый вход SPI
4	**PORTB4**	–	**PB4**	$\overline{SS}$ – вход выбора ведомого SPI
3	**PORTB3**	Выбор **HG2** (правого, младшего индикатора)	**PB3**	AIN1 – отрицательный вывод компаратора
2	**PORTB2**	Выбор **HG1** (левого, старшего индикатора)	**PB2**	AIN0 – положительный вывод компаратора
1	**PORTB1**	**SB5**	**PB1**	T1 – вход тактового сигнала таймера/счетчика 1
0	**PORTB0**	**SB4**	**PB0**	T0 – вход тактового сигнала таймера/счетчика 0

Бит	7	6	5	4	3	2	1	0
Имя бита	**DDB7**	**DDB6**	**DDB5**	**DDB4**	**DDB3**	**DDB2**	**DDB1**	**DDB0**

Рис. 20. Регистр направления данных порта B – **DDRB**. Адрес – $17($37)

Бит	7	6	5	4	3	2	1	0
Имя бита	**PINB7**	**PINB6**	**PINB5**	**PINB4**	**PINB3**	**PINB2**	**PINB1**	**PINB0**

Рис. 21. Регистр входных данных порта B – **PINB.** Адрес – $16($36)

Таблица 25

Регистр данных порта C – **PORTC.** Адрес – $15($35)

Бит	Имя бита	Подключенный элемент блока управления	Вывод порта	Альтернативная функция
7	**PORTC7**	HG1, HG1 – точка	**PC7**	**TOSC1** – таймера/счетчика 2
6	**PORTC6**	HG1, HG1 – сегмент **g**	**PC6**	**TOSC2** – таймера/счетчика 2
5	**PORTC5**	HG1, HG1 – сегмент **f**	**PC5**	Нет
4	**PORTC4**	HG1, HG1 – сегмент **e**	**PC4**	Нет
3	**PORTC3**	HG1, HG1 – сегмент **d**	**PC3**	Нет
2	**PORTC2**	HG1, HG1 – сегмент **c**	**PC2**	Нет
1	**PORTC1**	HG1, HG1 – сегмент **b**	**PC1**	Нет
0	**PORTC0**	HG1, HG1 – сегмент **a**	**PC0**	Нет

Бит	7	6	5	4	3	2	1	0
Имя бита	**DDC7**	**DDC6**	**DDC5**	**DDC4**	**DDC3**	**DDC2**	**DDC1**	**DDC0**

Рис. 22. Регистр направления данных порта C – **DDRC**. Адрес – $14($34)

Бит	7	6	5	4	3	2	1	0
Имя бита	**PINC7**	**PINC6**	**PINC5**	**PINC4**	**PINC3**	**PINC2**	**PINC1**	**PINC0**

Рис. 23. Регистр входных данных порта C – **PINC.** Адрес – $13($33)

У порта C только два вывода могут выполнять альтернативные функции: выводы **PC6** и **PC7** выполняют функции **TOSC1** и **TOSC2** таймера/счетчика 2.

Таблица 26

Регистр данных порта D – **PORTD.** Адрес – $12($32)

Бит	Имя бита	Подключенный элемент блока управления	Вывод порта	Альтернативная функция
7	**PORTD7**	**HA1** звукоизлучатель	**PD7**	OC2 – вывод сравнения выхода таймера/счетчика 2
6	**PORTD6**	**VD6** красный	**PD6**	ICP – вход триггера захвата таймера/счетчика 1
5	**PORTD5**	**VD5** желтый	**PD5**	OC1A – вывод сравнения выхода A таймера/счетчика 1
4	**PORTD4**	**VD4** зеленый	**PD4**	OC1B – вывод сравнения выхода B таймера/счетчика 1
3	**PORTD3**	**SA5**	**PD3**	**INT1** – вход внешнего прерывания 1
2	**PORTD2**	**SA4**	**PD2**	**INT0** – вход внешнего прерывания 0
1	**PORTD1**	–	**PD1**	TxD – выход передатчика UART
0	**PORTD0**	–	**PD0**	RxD – вход приемника UART

Бит	7	6	5	4	3	2	1	0
Имя бита	**DDD7**	**DDD6**	**DDD5**	**DDD4**	**DDD3**	**DDD2**	**DDD1**	**DDD0**

Рис. 24. Регистр направления данных порта D – **DDRD**. Адрес – $11($31)

Бит	7	6	5	4	3	2	1	0
Имя бита	**PIND7**	**PIND6**	**PIND5**	**PIND4**	**PIND3**	**PIND2**	**PIND1**	**PIND0**

Рис. 25. Регистр входных данных порта D – **PIND**. Адрес – $10($30)

Варианты индивидуальных заданий

1. Составьте программу вычитания из числа 4 числа 3. Если включен тумблер SA2, то на индикатор HG1 должен выдаваться результат вычитания. Если тумблер SA2 отключен – на индикацию выводится буква Н.

2. Составьте программу вычитания из числа 6 числа 3. Если включен тумблер SA3, то на индикатор HG1 должен выдаваться результат вычитания. Если тумблер SA3 отключен – на индикацию выводится буква П.

3. Составьте программу вычитания из числа 6 числа 4. Если включен тумблер SA4, то на индикатор HG1 должен выдаваться результат вычитания. Если тумблер SA4 отключен – на индикацию выводится буква Н.

4. Составьте программу вычитания из числа 7 числа 4. Если включен тумблер SA5, то на индикатор HG1 должен выдаваться результат вычитания. Если тумблер SA5 отключен – на индикацию выводится буква П.

5. Составьте программу вычитания из числа 7 числа 5. Если нажата кнопка SB1, то на индикатор HG2 должен выдаваться результат вычитания. Если кнопка SB1 отключена – на индикацию выводится буква Н.

6. Составьте программу вычитания из числа 8 числа 5. Если нажата кнопка SB2, то на индикатор HG2 должен выдаваться результат вычитания. Если кнопка SB2 отключена – на индикацию выводится буква П.

7. Составьте программу вычитания из числа 8 числа 4. Если нажата кнопка SB3, то на индикатор HG2 должен выдаваться результат вычитания. Если кнопка SB3 отключена – на индикацию выводится буква Н.

8. Составьте программу вычитания из числа 9 числа 4. Если нажата кнопка SB4, то на индикатор HG2 должен выдаваться результат вычитания. Если кнопка SB4 отключена – на индикацию выводится буква П.

Контрольные вопросы

1. Опишите состав лабораторного комплекса «Микроконтроллеры и автоматизация».

2. В чем заключается назначение программного продукта AVR Studio?

3. Расскажите о системе команд программируемого микроконтроллера AT90S8535.

4. Объясните, в чем заключается отладка программы.

ЧАСТЬ 2. СИСТЕМА ПАРАЛЛЕЛЬНОГО ВВОДА/ВЫВОДА

Цель работы

Изучение системы команд программируемого микроконтроллера AT90S8535, подготовка программы, содержащей подпрограммы, ее отладка, запись в микроконтроллер и демонстрация работы подготовленной программы.

Пояснения к работе

Работа является логическим продолжением лабораторной работы № 1. Задачи здесь усложнены, увеличено количество элементов ввода и вывода информации. Кроме того, управление индикацией светодиодов и семисегментных индикаторов рекомендуется выполнить в виде подпрограммы. Тогда их можно будет использовать при выполнении других лабораторных работ.

Для вызова подпрограммы используется команда RCALL pprmet, где pprmet – это метка и название подпрограммы.

Подпрограмма начинается с метки (pprmet), затем следует текст подпрограммы, а в конце подпрограммы ставится команда возврата из подпрограммы RET. Команда RET возвращает управление на команду, следующую за командой вызова подпрограммы RCALL pprmet.

Варианты индивидуальных заданий

1. Составьте программу сложения двух чисел и индикации младшей тетрады результата на семисегментном индикаторе HG2. На блоке управления располагаются кнопки и тумблеры. Кнопки SB2 и SB3 представляют собой младшие разряды первого слагаемого, т.е. возможен набор чисел от 0 до 3. Тумблеры SA4, SA5 представляют собой соответственно третий и четвертый разряды второго слагаемого, т.е. возможен набор десятичных чисел 0, 4, 8, 12. На индикаторе должен высвечиваться результат в шестнадцатеричном формате, т.е. 0, ..., F.

2. В памяти записаны два массива, например по пять ячеек в каждом. В ячейках записаны шестнадцатеричные цифры от 00 до 0FH. Цифры в массивах, кроме одной, разные. Составьте программу для определения, какая цифра присутствует в обоих массивах. При нажатии на кнопку SB3 (кратковременно) программа должна высвечивать результат в шестнадцатеричном виде на индикаторе HG2. Если цифры в массивах разные, то при нажатии на кнопку SB3 (кратковременно) на индикаторе HG2 должен высвечиваться символ «H» (нет).

3. Составьте программу управления индикацией таким образом, чтобы при нажатии на кнопку SB1 загорались цифра 1 и светодиод VD4, при нажатии на кнопку SB2 загорались цифра 2 и светодиод VD5, при одновременном нажатии кнопок SB1 и SB2 загоралась цифра 3 и включался светодиод VD6.

4. Составьте программу умножения положительного числа 2 на число 3. При нажатой кнопке SB1 на индикаторе HG2 программа должна высветить результат. При отпущенной кнопке на индикаторе должен гореть символ П.

5. Составьте программу управления индикацией. При включении микроконтроллера и кратковременном нажатии кнопки SB6 «RESET» при отключенном тумблере SA1 горит светодиод VD4 и на индикаторе HG1 горит цифра 2. При включении тумблера SA1 светодиод VD4 и индикатор HG1 гаснут, загорается светодиод VD5 и на индикаторе HG2 загорается цифра 5. При нажатой кнопке SB2 все светодиоды и индикаторы гаснут.

6. Составьте программу сложения двух чисел 3 и 4. При включении микроконтроллера и кратковременном нажатии кнопки SB6 «RESET» должен загораться и постоянно гореть светодиод VD5. При нажатой кнопке SB1 на индикаторе HG2 должно гореть первое слагаемое, при нажатой кнопке SB2 – второе слагаемое, при нажатой кнопке SB3 – результат.

7. В массиве из 16 ячеек памяти располагаются шестнадцатеричные числа от 00 до 00F. В массиве есть только одно число, которое повторяется несколько раз. Составьте программу для выявления этого числа и количества его повторов. При нажатии на кнопку SB1 на семисегментном индикаторе HG1 должно загораться повторяющееся число. При нажатии на кнопку SB2 на семисегментном индикаторе HG2 должно высвечиваться число его повторений. Если нет повторяющегося числа, то при нажатии на кнопку SB1 на индикаторе HG1 должен загораться символ «Н» (нет).

8. При включении микроконтроллера на индикаторе HG2 горит «0». Составьте программу для счета и индикации числа нажатий кнопки SB1 на индикаторе HG2 в шестнадцатеричном виде, т.е. 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2 3 4. Нажатие кнопки должно сопровождаться загоранием светодиода VD5. Для упрощения программы мер борьбы с дребезгом контактов не предпринимайте.

Контрольные вопросы

1. Что включают системы команд программируемого микроконтроллера AT90S8535?
2. Объясните, в чем заключается отличие программы от подпрограммы.
3. Поясните порядок записи программы в микроконтроллер.

ЧАСТЬ 3. ДИНАМИЧЕСКАЯ ИНДИКАЦИЯ

Цель работы

Освоение организации динамической индикации, используемой в программах, экспериментальное определение влияния длительности задержек на качество индикации.

Пояснения к работе

Работа продолжает изучение 8-разрядных двунаправленных портов программируемого микроконтроллера AT90S8535 для ввода и вывода дискретной информации, при этом используется динамическая индикация.

Ввод и вывод цифровых данных в микроконтроллерах семейства AVR и, в частности, микроконтроллере AT90S8535 могут осуществляться последовательно и параллельно. В рассматриваемой лабораторной работе речь будет идти об обмене информацией через двунаправленные параллельные порты A, B, C и D.

Управление портами, назначение и адресация их регистров уже указаны в описании к лабораторной работе № 1. Все рассматриваемые порты в качестве цифровых I/O портов общего назначения работают одинаково:

1. При нажатии на кнопку «Сброс» (RESET) осуществляется аппаратный сброс микроконтроллера и все каналы портов ставятся в третье состояние. На блоке управления рабочего места при этом горят светодиоды **VD4**–**VD6**, сегменты семисегментных индикаторов погашены.

2. Каждый вывод порта может быть запрограммирован индивидуально на ввод или вывод. Если в регистре направления данных порта для рассматриваемого бита записать «0», то соответствующий вывод конфигурируется как вход, при записи «1» – как выход.

3. Если вывод микроконтроллера сконфигурирован как вход, то при замыкании этого вывода на общий провод (⊥) вход воспринимается как «0». Неподключенный вывод воспринимается как наличие входного сигнала «1», уровень напряжения на выводе неподключенного входа составляет 3,5–4 В. Однако при этом повышается влияние помех, особенно проявляется влияние дребезга контактов. Уменьшить это влияние можно подключением вывода входного сигнала на питающее напряжение +5 В.

4. Реально уровень напряжения на выводе, определенном как выход, составляет при логической «1» – 4,5–5 В, при логическом «0» – 0–0,5 В.

5. Все выводы портов незапрограммированного микроконтроллера находятся в третьем состоянии.

В блоке управления используются семисегментные индикаторы HG1 и HG2 с общим катодом. Для замыкания катодов индикаторов на общий провод служат транзисторы VT1 и VT2, которые управляются соответственно битами **PB2** и **PB3** порта **B**. Транзисторы VT1 и VT2 при соответствующей программе обеспечивают **динамическую индикацию информации** на семисегментных индикаторах, т.е. вывод различной информации на оба индикатора. В этом случае поочередно зажигается то первый, то второй индикатор. При достаточно большой частоте переключений создается иллюзия одновременного непрерывного свечения индикаторов.

Для зажигания информации на первом индикаторе необходимо установить в «1» бит **PB2** и записать в порт **C** код выводимого первого символа. Затем, через некоторую программно реализуемую задержку времени, нужно сбросить бит **PB2**, установить в «1» бит **PB3** и подать на порт **C** код второго выводимого символа. Через задержку времени нужно сбросить бит **PB3**, установить в «1» бит **PB2** и т.д.

В качестве подпрограммы рекомендуется оформить временную задержку. Длительность задержки регулируется либо изменением величин констант (255), либо изменением значения переменной. Изменения значения переменной ust1 осуществляются в основной программе:

```
wait3:                  ;подпрограмма временной задержки
lds   r18,ust1          ;задание величины задержки третьей ступени
w0:
ldi   r19,255           ;задание величины задержки второй ступени
w1:
ldi   r20,255           ;задание величины задержки первой ступени
w2:
dec   r20                       ;временная задержка третьей ступени
brne  w2
dec   r19                       ;временная задержка второй ступени
brne  w1
dec   r18                       ;временная задержка первой ступени
brne  w0
ret                             ;возврат из подпрограммы
```

Если нет необходимости в длительной задержке, количество ступеней в подпрограмме можно уменьшить.

Варианты индивидуальных заданий

1. Составьте программу, реализующую последовательное формирование свечения цифры 3 на индикаторе HG1. При подаче питания и нажатии на кнопку SB6 «Сброс» все светодиоды и индикаторы погашены. При включении тумблера SA1 включаются сегменты индикатора HG1 в последовательности a, b, c, d, g. После этого раздается щелчок звукогенератора HA1. На индикаторе горит цифра 3. При отключении тумблера SA1 гаснут сегменты индикатора HG1 в последовательности g, d, c, b, a. Включение тумблера SA2 изменяет темп формирования цифры 3.

2. Составьте программу организации счета числа нажатий кнопки SB4. При подаче питания и нажатии кнопки SB6 «Сброс» все светодиоды и индикаторы погашены. При каждом очередном нажатии кнопки SB4 число увеличивается. После 10 нажатий кнопки загорается светодиод VD4, на индикаторе HG1 загорается цифра 1 и раздается щелчок звукогенератора HA1. После второго десятка нажатий кнопки дополнительно загорается светодиод VD5, на индикаторе загорается цифра 2 и раздается щелчок звукогенератора. Дальнейшие нажатия кнопки SB4 не меняют состояния схемы. Нажатие кнопки SB5 гасит все светодиоды и индикатор и схема приходят в исходное состояние.

3. Составьте программу организации счета числа нажатий кнопок SB4 и SB5. При подаче питания и нажатии кнопки SB6 «Сброс» на индикаторах HG2 и HG1 горит число 00. При каждом очередном нажатии на кнопку SB4 число на индикаторах увеличивается на единицу. Счет возможен до 20. Если счет достиг числа 20, то дальнейшие нажатия кнопки SB4 число не меняют. При каждом очередном нажатии на кнопку SB5 число на индикаторах уменьшается на единицу. При достижении числа 00 дальнейшие нажатия кнопки SB5 не влияют на схему. Нажатие кнопки SB1 обнуляет индикаторы.

4. Составьте программу организации «бегущего» огня по сегментам семисегментных индикаторов HG1 и HG2. При «беге» по часовой стрелке чередование сегментов следующее: a, b, c, d, e, f, a, b… и т.д. При включенном тумблере SA1 реализуется «бегущий» огонь по часовой стрелке, при отключенном – против часовой стрелки. При включенном тумблере SA2 «бегущий» огонь реализуется по сегментам индикатора HG1, при отключенном – по сегментам индикатора HG2.

5. Реализуйте на микроконтроллере схему управления светофором. При включении тумблера SA1 светофор работает в дневном режиме, т.е. чередование сигналов следующее: зеленый (VD4), желтый (VD5), красный

(VD6), желтый, зеленый, желтый и т.д. При отключении тумблера SA1 светофор работает в ночном режиме, т.е. мигает желтый светодиод VD5. В дневном режиме работы на индикаторе HG1 горит буква «d», в ночном режиме – буква «H».

6. Составьте программу управления индикацией. При подаче питания и нажатии на кнопку SB6 «Сброс» загорается светодиод VD4, а на индикаторах HG1 и HG2 горит число 04, т.е. номер светодиода. При нажатии и отпускании кнопки SB2 светодиод VD4 гаснет, а VD5 загорается, т.е. происходит сдвиг свечения вправо. На индикаторах загорается число 05. При каждом очередном нажатии на кнопку SB2 свечение сдвигается вправо, т.е. наблюдается свечение VD4, VD5, VD6, VD4, VD5… и т.д. При этом на индикаторах высвечиваются соответственно числа 04, 05, 06, 04, 05… и т.д. При нажатии и отпускании кнопки SB3 схема работает аналогично, но сдвиг свечения происходит влево.

7. Составьте программу управления индикацией. При подаче питания и нажатии кнопки SB6 «Сброс» загораются светодиоды VD4 и VD6 и на индикаторах HG1 и HG2 соответственно горят цифры 4 и 6. При нажатии и удержании кнопки SB2 светодиоды VD4 и VD6 гаснут, загорается светодиод VD5 и на индикаторах HG1 и HG2 соответственно загораются цифры 0 и 5. При отпускании кнопки SB2 схема приходит в исходное состояние.

8. Составьте программу управления индикацией. При подаче питания и нажатии кнопки SB6 «Сброс» на индикаторах HG1 и HG2 загораются соответственно цифры 7 и 5. При нажатии и отпускании кнопки SB2 на индикаторах загораются цифры 5 и 7 и включаются светодиоды VD4 и VD6. При нажатии и удержании кнопки SB3 все индикаторы и светодиоды гаснут, при отпускании возобновляется горение, как после нажатия кнопки SB6 «Сброс».

Контрольные вопросы

1. Расскажите, в чем заключается организация динамической индикации.
2. Объясните, как используется динамическая индикация в программах.
3. Поясните, как влияет длительность задержек на качество индикации.

ЧАСТЬ 4. СИСТЕМА ВНЕШНИХ ПРЕРЫВАНИЙ МИКРОКОНТРОЛЛЕРА

Цель работы

Изучение организации прерываний в микроконтроллере AT90S8535 и обслуживания подсистемы внешних прерываний.

Пояснения к работе

Микроконтроллер AT90S8535 использует 17 источников прерывания. Эти прерывания располагают отдельными векторами в пространстве памяти программ. Каждому прерыванию присвоен свой бит разрешения, который должен быть установлен совместно с битом **I** регистра статуса **SREG**.

Младшие адреса пространства памяти автоматически определяются как векторы сброса и прерываний.

Полный перечень векторов прерывания представлен в табл. 27.

Таблица 27

Векторы прерывания

Номер вектора	Адрес вектора	Причина	Формулировка прерываний
1	2	3	4
1	$000	RESET	Сброс по выводу RESET и сторожевому таймеру (Hardware Pin, Power-On Reset и 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/счетчика 2 (Timer/Conter2 Overflow)
6	$005	TIMER1 CAPT	Захват таймера/счетчика 1 (Timer/Conter1 Capture Event)
7	$006	TIMER1 COMPA	Совпадение А при сравнении таймера/счетчика 1 (Timer/Conter1 Compare Match A)

Окончание табл. 27

1	2	3	4
8	$007	TIMER1 COMPB	Совпадение B при сравнении таймера/счетчика 1 (Timer/Conter1 Compare Match B)
9	$008	TIMER1 OVF	Переполнение таймера/счетчика 1 (Timer/Conter1 Overflow)
10	$009	TIMER0 OVF	Переполнение таймера/счетчика 0 (Timer/Conter0 Overflow)
11	$00A	SPI, STC	Завершение пересылки SPI (SPI Serial Transfer Complete)
12	$00B	UART, RX	Завершение приема UART (UART, Rx Complete)
13	$00C	UART, UDRE	Регистр данных UART пуст (UART Data Register Empty)
14	$00D	UART, TX	Завершение передачи UART (UART, Tx Complete)
15	$00E	ADC	Завершение ADC преобразования (ADC Conversion Complete)
16	$00F	EE_RDY	Готовность EEPROM (EEPROM Ready)
17	$010	ANA_COMP	Срабатывание аналогового компаратора (Analog Comparator)

Прерывания с младшими адресами имеют больший уровень приоритета. **RESET** имеет наивысший уровень приоритета, следующим является запрос внешнего прерывания **INT0** и т.д.

Микроконтроллеры AT90S8535 содержат специальный восьмиразрядный регистр масок внешних прерываний (рис. 26).

Бит	7	6	5	4	3	2	1	0
Имя бита	**INT1**	**INT0**	—	—	—	—	—	—
Чтение/Запись	R/W	R/W	R	R	R	R	R	R
Исходное значение	0	0	0	0	0	0	0	0

Рис. 26. Регистр масок внешнего прерывания – GIMSK. Адрес – $3B($5B)

Биты 7 и 6 – **INT1, INT0**: разрешение внешних прерываний соответственно **INT1** и **INT0**. При установленных битах **INT1**, **INT0** и установленном бите **I** регистра статуса (**SREG**) разрешаются прерывания по соответствующим выводам внешних прерываний.

Биты 5–0: зарезервированные биты. Эти при считывании всегда покажут состояние 0.

Активизация выводов **INT1** и **INT0** вызывает запрос прерывания, если даже эти выводы будут разрешены как выход.

Условия возникновения запроса (по нарастающему/спадающему фронту сигнала или по логическому уровню) задаются с помощью битов управления опознанием прерывания регистра **MCUCR** (рис. 27).

Бит	7	6	5	4	3	2	1	0
$35($55)	**—**	SE	SM1	SM0	**ISC11**	**ISC10**	**ISC01**	**ISC00**
Чтение/Запись	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Исходное значение	0	0	0	0	0	0	0	0

Рис. 27. Регистр контроля (управления) **– MCUCR**. Адрес – $35($55)

Этот регистр содержит служебные биты для общих функций **MCU**.

Бит 7 – этот бит зарезервирован в AT90S8535 и при считывании всегда показывает состояние ноль.

Биты 6–4 – управляют режимами энергосбережения.

Биты 3, 2 – ISC11, ISC10: биты управления идентификацией внешнего прерывания 1. Внешнее прерывание 1 активизируется внешним выводом **INT1**, если установлен флаг **I** в регистре статуса **SREG** и установлена соответствующая маска прерывания в регистре масок внешнего прерывания **GIMSK** (**INTF1=1**).

Запрос прерывания по логическому уровню или фронтам определяется в табл. 28.

Таблица 28

Задание характера сигнала прерывания 1

ISC11	ISC10	Описание
0	0	Запрос прерывания идентифицируется по низкому уровню на INT1
0	1	Зарезервирован
1	0	Запрос прерывания идентифицируется по спадающему фронту на INT1
1	1	Запрос прерывания идентифицируется по нарастающему фронту на INT1

Примечание: *При программном изменении битов* ***ISC11/ISC10*** *прерывание* ***INT1*** *должно быть запрещено путем очистки бита разрешения прерывания в регистре* ***GIMSK****. В ином случае может произойти прерывание.*

Биты 1, 0 – ISC01, ISC00: биты управления идентификацией внешнего прерывания 0. Внешнее прерывание 0 активизируется внешним выводом **INT0**, если установлен флаг **I** в регистре статуса **SREG** и установлена соответствующая маска прерывания. Запрос прерывания по логическому уровню или фронтам определяется в табл. 29.

Таблица 29

Задание характера сигнала прерывания 0

ISC01	ISC00	Описание
0	0	Запрос прерывания идентифицируется по низкому уровню на INT0
0	1	Зарезервирован
1	0	Запрос прерывания идентифицируется по спадающему фронту на INT0
1	1	Запрос прерывания идентифицируется по нарастающему фронту на INT0

Примечание: *При программном изменении битов* ***ISC01/ISC00*** *прерывание* ***INT0*** *должно быть запрещено путем очистки бита разрешения прерывания в регистре* ***GIMSK****. В ином случае может произойти прерывание.*

Прерывания по уровню сигнала флага не имеют, и условия прерывания имеют место, пока активен внешний сигнал (рис. 28).

Бит	7	6	5	4	3	2	1	0
Имя бита	**INTF1**	**INTF0**	**—**	**—**	**—**	**—**	**—**	**—**
Чтение/Запись	R/W	R/W	R	R	R	R	R	R
Исходное значение	0	0	0	0	0	0	0	0

Рис. 28. Регистр флагов внешних прерываний **– GIFR.** Адрес – $3A($5A)

Биты 7 и 6 – INTF1, INTF0: флаги внешних прерываний **INTF1** и **INTF0.** В случае поступления запроса на прерывание на какой-либо из указанных выводов устанавливается (=1) соответствующий флаг прерывания. Если бит **I** регистра **SREG** и соответствующий бит разрешения регистра **GIMSK** установлены, то выполняется переход по вектору прерывания. При возврате из процедуры прерывания флаг очищается. Кроме того, флаг можно очистить, записав в него логическую 1.

Биты 5–0 – эти биты зарезервированы в AT90S8535 и при считывании всегда покажут состояние 0.

При возникновении прерывания бит **I** разрешения глобального прерывания (*Global Interrupt Enable*) очищается и все прочие прерывания запрещаются.

Для разрешения вложенных прерываний необходимо установить бит **I** внутри подпрограммы обработки прерывания. Выход из подпрограммы обработки прерывания происходит по команде **RETI**, при этом бит **I** устанавливается в состояние **1**. Когда счетчик команд указывает вектор подпрограммы обработки прерывания, соответствующий флаг, вызвавший прерывание, аппаратно очищается. Некоторые флаги прерываний можно очистить, записав в соответствующий бит(ы) очищаемого флага логическую единицу.

Если условия прерываний возникли, когда очищен бит разрешения всех прерываний, поступающие прерывания устанавливают свои флаги, и они будут сохранены в таком состоянии, пока не возникает разрешение глобального прерывания, и будут обработаны в порядке приоритетов.

Обратите внимание! Регистр состояния автоматически не сохраняется при закрытии программы, т.е. при открытии прерывающей подпрограммы он должен быть сохранен и должен быть восстановлен при возвращении из прерывающей подпрограммы. Выполнение этого условия должно осуществляться за счет учета в программе.

При проведении лабораторной работы записывается фоновая программа, которая будет прерываться внешними источниками **INT0** и **INT1**. Разрешение на прерывание, его приоритет и активный характер сигнала запроса прерывания задаются с тумблеров блока управления по усмотрению разработчиков программы.

В каждом варианте указано, какие внешние визуальные проявления должны происходить при прерывании. Каждый обучающийся при выполнении работы должен посмотреть реакцию системы при следующих ситуациях:

а) запретить все прерывания и убедиться, что включения и отключения тумблеров **INT0** и **INT1** не оказывают влияния на работу фоновой программы;

б) разрешить прерывания по перепаду для **INT0** и **INT1** и посмотреть реакцию микроконтроллера на эти прерывания;

в) убедиться, что прерывание по **INT0** прерывает выполнение программы по **INT1**;

г) проделать пункты б) и в) при задании прерывания по уровню и проанализировать, какие отличия имеют место при отработке прерываний по уровню от отработки прерываний по перепаду.

Ниже приведена структура одного из вариантов программы при выполнении лабораторной работы.

```
; Примерная структура программы процедуры обслуживания прерываний
.include "8535def.inc"
; ***Формирование таблицы переходов***
.org $000
rjmp start
.org $001               ; вектор int0
rjmp pr0
.org $002               ; вектор int1
rjmp pr1
.org $003               ; вектор Timer2 Compare
reti
.org $004               ; вектор Timer2 Overflow
reti
.org $005               ; вектор Timer1 Capture
reti
.org $006               ; вектор Timer1 CompareA
reti
.org $007               ; вектор Timer1 CompareB
reti
.org $008               ; вектор Timer1 Overflow
reti
.org $009               ; вектор Timer0 Overflow
reti
.org $00A               ; вектор SPI
reti
.org $00B               ; вектор приемника UART
reti
.org $00C               ; вектор "буфер передатчика пуст" UART
reti
.org $00D               ; вектор передатчика UART
reti
.org $00E               ; вектор ADC преобразователя
reti
.org $00F               ; вектор EEPROM
reti
```

```
        .org $010                ; вектор аналогового компаратора
        reti
        ; ***Фоновая программа***
        .cseg
        .org $030
start:  ldi   r16,$00            ; загрузка указателя стека через регистр r16
        out   sph,r16
        ldi   r16,$0ff
        out   spl,r16
m1:
        sbis  PINA,1             ; опросить кнопку SB1. От ее состояния
                                 зависит
        rjmp  m1
        sei                      ; разрешение глобального прерывания
        ldi   r16,$c0
        out   GIMSK,r16          ; разрешение прерываний INT0 и INT1
        ldi   r16,$f0            ; выводы PD0, PD1, PD2 и PD3
                                 конфигурировать как
        out   DDRD,r16           ; входы, а остальные выводы порта D – как
                                 выходы

        ldi   r16,$0ff           ; выводы порта C конфигурировать как
                                 выходы
        out   DDRC,r16

        sbi   PORTB,3            ; загрузить 1 в бит portB,3 для включения
                                 ; транзистора VT2

        ***                      ;команды фоновой программы по
                                 индивидуальному
                                 ; заданию
        rcall wait
        in    r16,PINA           ; опросить кнопки и тумблеры порта A
        lsr   r16                ; сдвинуть разряды регистра r16 три раза
                                 вправо
        lsr   r16
        lsr   r16
        ldi   r17,$0f
        and   r16,r17            ; биты ISC01, ISC00, ISC11, ISC10
                                 принимают
```

```
        out     MCUCR,r16       ; соответственно состояние SB1, SA1, SA2,
                                  SA3

pr0:                            ; Подпрограмма отработки прерывания
                                  INT0
        in      r16,SREG        ; сохранение SREG
        push    r16

        ***                     ; команды, реализующие индивидуальное
                                  задание

        cbi     GIFR,6          ; сброс флага прерывания по INT0
        pop     r16             ; восстановление SREG
        out     SREG,r16
        reti

pr1:                            ; Подпрограмма отработки прерывания
                                  INT1
        in      r16,SREG        ; сохранение SREG
        push    r16

        ***                     ; команды, реализующие индивидуальное
                                  задание

        cbi     GIFR,7          ; сброс флага прерывания по INT1
        pop     r16             ; восстановление SREG
        out     SREG,r16
        reti

wait:                           ; подпрограмма задержки
        ldi     r20,$0ff
loop1:
        ldi     r21,$0ff
loop:
        dec     r21
        brne    loop
        dec     r20
        brne    loop1
        ret
```

Представленная программа дает пример формирования таблицы переходов и определения начального адреса стека. Кнопки и тумблеры блока управления рабочего места используются для вмешательства в программу микроконтроллера:

– нажатие кнопки **SB1** разрешает глобальное прерывание;

– кнопка **SB3** и тумблер **SA1** обеспечивают задание характера сигнала прерывания **INT0** (по фронтам или по уровню сигнала);

– тумблеры **SA2** и **SA3** аналогично обеспечивают задание характера сигнала прерывания **INT1.**

В представленной программе подпрограмма **wait** реализует задержку времени. Максимальная длительность задержки будет при записи в регистры **r20** и **r21** десятичных чисел **255**. При необходимости увеличения длительности задержки можно реализовать несколько команд типа **wait** (можно с разными уставками времени) или выполнять последовательно несколько раз одну и ту же подпрограмму задержки времени. Можно записать свою подпрограмму **wait**, используя большее число регистров.

В индивидуальных заданиях есть необходимость изменения скорости «бегущих» огней. Это можно обеспечить инкрементом-декрементом или сложением-вычитанием чисел в каких-либо ячейках или регистрах, которые могут передаваться в регистры **r20** и **r21** подпрограммы **wait.**

Варианты индивидуальных заданий

1. Составьте программу управления индикацией. При подаче питания и нажатии на кнопку SB6 «Сброс» загорается светодиод VD4. По прерыванию **INT0** светодиод VD4 гаснет, загорается светодиод VD5 (идет сдвиг вправо). То есть по прерыванию **INT0** последовательно загораются и гаснут светодиоды VD4, VD5, VD6, VD4, VD5 и т.д. По прерыванию **INT1** сдвиг осуществляется влево.

2. Составьте фоновую программу управления индикацией, реализующую «бегущий» огонь на светодиодах VD4, VD5, VD6 слева направо, т.е. поочередно загораются светодиоды VD4, VD5, VD6, VD4, VD5 и т.д. По прерыванию **INT0** скорость «бега» увеличивается, а по прерыванию **INT1** – уменьшается.

3. Составьте фоновую программу управления индикацией, реализующую «бегущий» огонь на сегментах индикатора HG2, т.е. последовательно загораются и гаснут сегменты a, b, c, d, e, f, a, b, c и т.д. Прерывание по **INT0** изменяет направление «бега» на противоположное. Прерывание по **INT1** восстанавливает исходное направление «бега».

4. Составьте программу управления индикацией. При подаче питания и нажатии кнопки SB6 «Сброс» на индикаторе HG2 загорается цифра 0. По прерыванию **INT0** высвечиваемая цифра увеличивается на единицу, по прерыванию **INT1** – уменьшается на единицу. Диапазон изменения цифры – от 0 до 9. Если цифра достигла значения 9, то дальнейшие прерывания **INT0** не меняют ее. Если же высвечивается цифра 0, то прерывания **INT1** не должны изменять эту цифру.

5. Составьте фоновую программу, реализующую работу светофора, т.е. следующее чередование сигналов: зеленый (VD4), желтый (VD5), красный (VD6), желтый, зеленый и т.д. По прерыванию **INT0** время горения каждой из ламп увеличивается, а по прерыванию **INT1** – уменьшается.

6. Составьте программу управления индикацией. При подаче питания и нажатии кнопки SB6 «Сброс» на индикаторе HG2 горит цифра 8. По прерыванию **INT0** число на индикаторе увеличивается на две единицы, по прерыванию **INT1** – уменьшается на две единицы. При достижении цифр 0 и E прерывания соответственно **INT0** и **INT1** не влияют на работу.

7. Составьте фоновую программу, при включении микроконтроллера обеспечивающую вывод на порт C комбинации сигналов, соответствующей индикации на индикаторе HG2 цифры 5. По прерыванию **INT0** состояние линии **PortC,1** инвертируется. По прерыванию **INT1** содержимое порта **C** инкрементируется. В отчете отразите получившиеся изображения на индикаторе.

8. Составьте программу управления индикацией. При включении микроконтроллера и нажатии на кнопку SB6 «Сброс» на порт **D** выдается число 20h, на порт **C** – число 66h. По прерыванию **INT0** производится циклический сдвиг регистра **PortD** вправо, по прерыванию **INT1** – влево.

Контрольные вопросы

1. Объясните, как организуются прерывания в микроконтроллере AT90S8535.

2. Поясните, в чем заключается обслуживание подсистемы внешних прерываний.

ЧАСТЬ 5. ПРОГРАММИРОВАНИЕ ТАЙМЕРОВ/СЧЕТЧИКОВ

Цель работы

Изучение функционирования таймеров/счетчиков микроконтроллеров AT90S8535, получение практических навыков в их программировании.

Пояснения к работе

Периферийный модуль микроконтроллера AT90S8535 имеет три встроенных таймера/счетчика общего назначения – TC0, TC1 и TC2:

– TC0 – 8 разрядов;

– TC1 – 16 разрядов;

– TC2 – 8 разрядов.

Таймеры/счетчики TC0 и TC2 представляют собой 8-разрядные двоичные счетчики (рис. 29) с возможностью предварительной записи информации и ее считывания.

С точки зрения обмена информацией счетчики ничем не отличаются от регистров и обозначаются как регистры **TCNT0** и **TCNT2** соответственно.

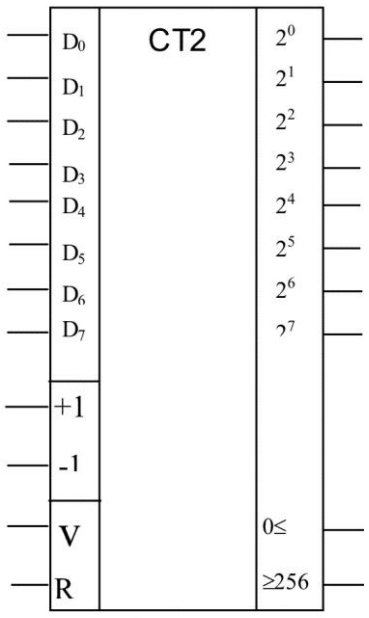


Рис. 29. Восьмиразрядный двоичный счетчик

Шестнадцатиразрядный таймер/счетчик TC1 представляет собой два последовательно соединенных 8-разрядных счетчика **TCNT1L** и **TCNT1H**. Так как шина данных 8-разрядная, то запись и считывание информации производится последовательно.

Таймеры/счетчики могут работать в следующих режимах:

– счетчика внешних событий (с внешним тактированием);

– таймера (от внутреннего источника опорной частоты);

– широтно-импульсного модулятора (ШИМ) цифрового сигнала.

Для организации работы таймеров/счетчиков, приема и передачи информации служит набор регистров пространства ввода/вывода. В табл. 30 перечислены регистры, общие для всех таймеров/счетчиков, а в табл. 31–33 приведены регистры, относящиеся к таймерам/счетчикам TC0, TC1 и TC2.

Таблица 30

Регистры обслуживания прерываний и фиксации событий для всех таймеров/счетчиков

Адрес I/O (адрес SRAM)	Обозначение	Функция
$39($59)	TIMSK	Регистр масок таймера счетчика
$38($58)	TIFR	Регистр флагов таймера счетчика

Таблица 31

Регистры таймера/счетчика TC0

Адрес I/O (адрес SRAM)	Обозначение	Функция
$33($53)	TCCR0	Регистр управления таймера/счетчика TC0
$32($52)	TCNT0	Таймер/счетчик TC0 8-бит

Таблица 32

Регистры таймера/счетчика TC1

Адрес I/O (адрес SRAM)	Обозначение	Функция
$2F($4F)	TCCR1A	Регистр управления таймера/счетчика TC1 A
$2E($4E)	TCCR1B	Регистр управления таймера/счетчика TC1 B
$2D($4D)	TCNT1H	Таймер/счетчик TC1, старший байт
$2C($4C)	TCNT1L	Таймер/счетчик TC1, младший байт
$2B($4B)	OCR1AH	Регистр сравнения таймера/счетчика TC1 A, старший байт
$2A($4A)	OCR1AL	Регистр сравнения таймера/счетчика TC1 A, младший байт
$29($49)	OCR1BH	Регистр сравнения таймера/счетчика TC1 B, старший байт
$28($48)	OCR1BL	Регистр сравнения таймера/счетчика TC1 B, младший байт
$27($47)	ICR1H	Регистр захвата таймера/счетчика TC1, старший байт
$26($46)	ICR1L	Регистр захвата таймера/счетчика TC1, младший байт

Таблица 33

Регистры таймера/счетчика TC2

Адрес I/O (адрес SRAM)	Обозначение	Функция
$25($45)	TCCR2	Регистр управления таймера/счетчика TC2
$24($44)	TCNT2	Таймер/счетчик TC2 8-бит
$23($43)	OCR2	Регистр сравнения таймера/счетчика TC2
$22($42)	ASSR	Регистр статуса асинхронного режима

Информация в эти регистры может записываться и считываться программными средствами и в дальнейшем выводиться через один из четырех портов ввода/вывода.

Непосредственный обмен информацией с внешней средой организуется через отдельные выводы портов, перечисленные в табл. 34.

Таблица 34

Выводы портов для обмена информацией с внешней средой

Порт	Вывод порта	Сигнал	Функции выводов порта
PORTB	PB0	T0	Вход асинхронного тактового сигнала таймера/счетчика 0
PORTB	PB1	T1	Вход асинхронного тактового сигнала таймера/счетчика 1
PORTC	PC6	TOSC1	Асинхронный тактовый сигнал таймера/счетчика 2 от внешнего генератора (с кварцевым кристаллом с частотой 32,768 кГц)
PORTC	PC7	TOSC2	Второй вход асинхронного тактового сигнала таймера/счетчика 2
PORTD	PD4	OC1B	Вывод сравнения выхода B таймера/счетчика 1
PORTD	PD5	OC1A	Вывод сравнения выхода A таймера/счетчика 1
PORTD	PD6	ICP	Вход триггера захвата таймера/счетчика 1
PORTD	PD7	OC2	Вывод сравнения выхода таймера/счетчика 2

Работа в режиме счетчика. В режиме счетчика таймеры/счетчики могут осуществлять счет сигналов от внешних источников, импульсов тактовой частоты микроконтроллера (СК) или импульсов с выходов делителей частоты.

Таймеры/счетчики TC0 и TC1 используют выходы ступеней деления общего 10-разрядного предварительного делителя (рис. 30).

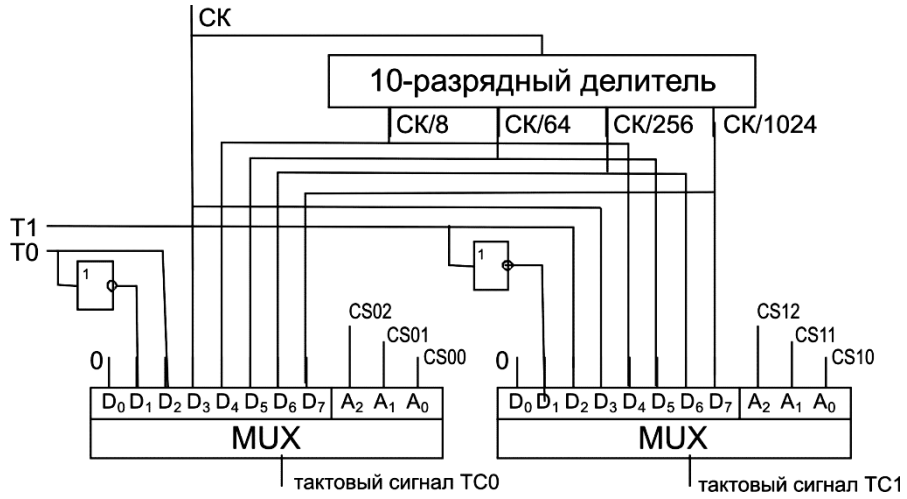


Рис. 30. Предварительные делители таймеров/счетчиков 0 и 1

Выбор источников тактовых сигналов обеспечивается комбинацией значений битов, установленных в регистрах управления **TCCR0** и **TCCR1B** соответственно.

Предварительный делитель таймеров/счетчиков 0 и 1 содержит четыре ступени деления: СК/8, СК/64, СК/256 и СК/1024, где **СК** – входной тактовый сигнал. Кроме того, в качестве источников тактовых сигналов могут быть использованы сигналы от внешних источников, тактовый сигнал **СК** и нулевой тактовый сигнал (**stop**), которые также могут быть использованы как часы.

Таймер/счетчик ТС2 имеет собственный 10-разрядный предварительный делитель (рис. 31).

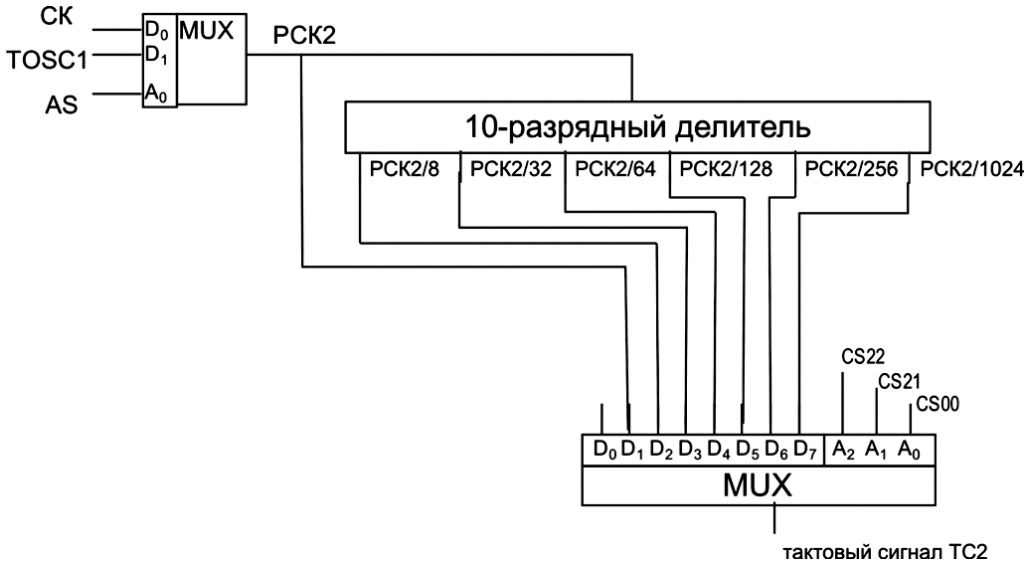


Рис. 31. Предварительный делитель таймера/счетчика 2

Тактовый сигнал таймера/счетчика 2 обозначен **РСК2**. Этот тактовый сигнал по умолчанию подключен к основному тактовому сигналу системы **СК**. При установке бита **AS2** в регистре **ASSR** таймер/счетчик 2 будет асинхронно тактироваться сигналом с вывода **PC6** (**TOSC1**), что позволяет использовать таймер/счетчик 2 в качестве часов реального времени (**RTC**), когда **AS2** установлен таким образом, что соединяет **PC6** (**TOSC1**) и **PC7** (**TOSC2**) через разъединительный порт **C**.

Генератор оптимизирован под использование кварцевого кристалла с частотой 32,768 кГц, подсоединяемого между выводами **PC6** (**TOSC1**) и **PC7** (**TOSC2**). При тактировании таймера/счетчика 2 внешним тактовым сигналом этот сигнал синхронизируется с тактовой частотой CPU. Для обеспечения правильной синхронизации внешнего сигнала необходимо, чтобы минимальное время между двумя входящими тактовыми циклами было не менее одного цикла внутреннего тактового сигнала CPU. Внешний тактовый сигнал синхронизируется нарастающим фронтом внутреннего тактового сигнала CPU.

Частота внешнего тактового сигнала, подаваемого на вывод **TOSC1**, не должна превышать одной четвертой от тактовой частоты процессора, но не выше чем 256 кГц.

На рис. 32 показана блок-схема таймера/счетчика 0.

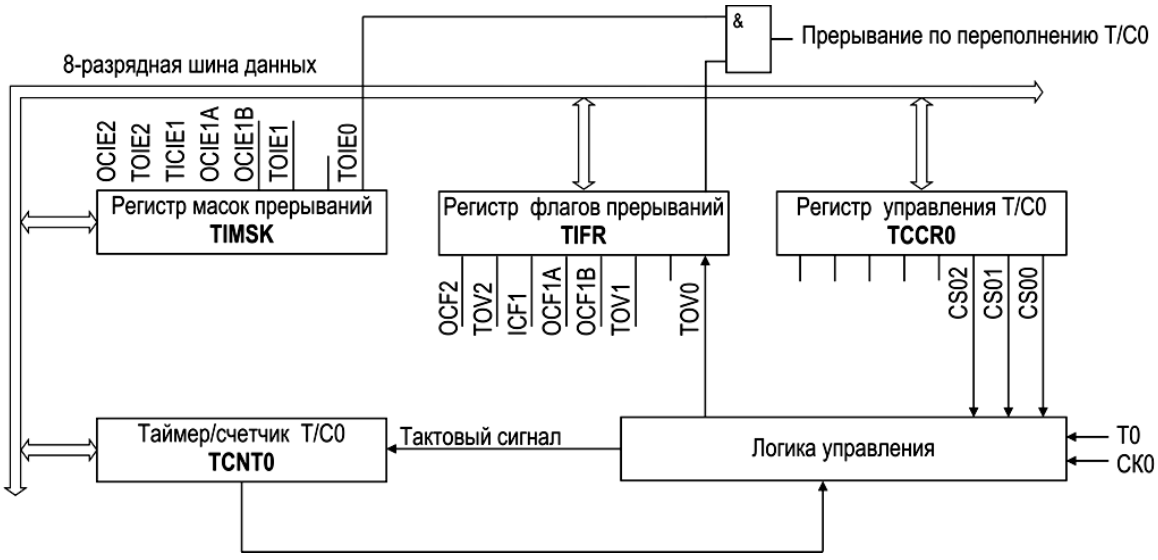


Рис. 32. Блок-схема таймера/счетчика 0

Восьмиразрядный таймер-счетчик 0 получает тактовый сигнал или непосредственно от СК, или от внешнего вывода.

Кроме этого, таймер-счетчик 0 может быть остановлен, как это показано в описании регистра управления Timer/Counter0 – **TCCR0**.

В регистре флагов прерывания таймера-счетчика 0 **TIFR** хранится флаг состояния переполнения.

Установки управляющих сигналов хранятся в регистре управления таймером/счетчиком – **TCCR0** (рис. 33). Установка разрешения/запрещения прерываний производится в регистре масок прерываний таймеров/счетчиков – **TIMSK**.

Бит	7	6	5	4	3	2	1	0
$33($53)	**—**	**—**	**—**	**—**	**—**	**CS02**	**CS01**	**CS00**
Чтение/Запись	R	R	R	R	R	R/W	R/W	R/W
Исходное значение	0	0	0	0	0	0	0	0

Рис. 33. Регистр управления таймером/счетчиком 0 – **TCCR0**
(The Timer/Counter0 Control Register)

Точность и разрешение 8-разрядных таймеров/счетчиков растет с уменьшением коэффициента предварительного деления. Аналогичным образом высокий коэффициент предварительного деления удобно использовать при реализации функций с низким быстродействием или при точной синхронизации редко происходящих действий.

Биты 2, 1 и **0** выбора тактовой частоты TC0 подключают выход определенной ступени предварительного делителя. Величину коэффициента деления частоты определяют комбинации значений битов, приведенные в табл. 35.

Таблица 35

Выбор коэффициента деления предварительного делителя частоты таймера/счетчика 0

CS02	CS01	CS00	Описание
0	**0**	**0**	Таймер/счетчик 0 остановлен
0	**0**	**1**	CK0
0	**0**	**0**	CK / 8
0	**1**	**1**	CK / 64
1	**0**	**0**	CK / 256
1	**0**	**1**	CK / 1024
1	**1**	**0**	Внешний вывод T0, падающий фронт
1	**1**	**1**	Внешний вывод T0, нарастающий фронт

Если таймер/счетчик 0 используется как счетчик, то вывод **T0** конфигурируется как вход.

Таймер/счетчик 0 осуществляет счет в регистре **TCNT0** (рис. 34).

Бит	7	6	5	4	3	2	1	0
$32($52)	**MSB**	**—**	**—**	**—**	**—**	**—**	**—**	**LSB**
Чтение/Запись	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Исходное значение	0	0	0	0	0	0	0	0

Рис. 34. Регистр счета таймера/счетчика 0 – **TCNT0**

Timer/Counter 0 реализован как счетчик с возможностью чтения/записи. Если в Timer/Counter 0 записано некоторое значение и выбран источник тактового сигнала, то он продолжит счет с записанного значения с тактовой частотой счетчика.

Установки управляющих сигналов таймера/счетчика 1 хранятся в регистре управления таймером/счетчиком – **TCCR1B** (рис. 35).

Бит	7	6	5	4	3	2	1	0
$2E($4E)	ICNC1	ICES1	**—**	**—**	CTC1	**CS12**	**CS11**	**CS10**
Чтение/Запись	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Исходное значение	0	0	0	0	0	0	0	0

Рис. 35. Регистр управления B таймера/счетчика 1 – **TCCR1B**

Биты 2, 1 и **0** выбора тактовой частоты TC1 подключают выход определенной ступени предварительного делителя. Величину коэффициента деления частоты определяют комбинации значений битов, приведенные в табл. 36.

Таблица 36

Выбор источника тактового сигнала таймера/счетчика 1

CS12	CS11	CS10	Описание
0	**0**	**0**	**Stop**-условие – таймер/счетчик 1 остановлен
0	**0**	**1**	CK
0	**1**	**0**	CK / 8
0	**1**	**1**	CK / 64
1	**0**	**0**	CK / 256
1	**0**	**1**	CK / 1024
1	**1**	**0**	Внешний тактирующий сигнал на выводе **T1**, нарастающий фронт
1	**1**	**1**	Внешний тактирующий сигнал на выводе **T1**, падающий фронт

Если таймер/счетчик 1 используется как счетчик, то вывод **T1** конфигурируется как вход. Таймер/счетчик 1 осуществляет счет в регистрах **TCNT1H** и **TCNT1L**.

Установки управляющих сигналов таймера/счетчика 2 хранятся в регистре управления таймером/счетчиком – **TCCR2** (рис. 36). Величину коэффициента деления частоты определяют комбинации значений битов **2, 1** и **0**, приведенные в табл. 37.

Бит	7	6	5	4	3	2	1	0
$25($45)	**—**	PWM2	COM21	COM20	CTC2	**CS22**	**CS21**	**CS20**
Чтение/Запись	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Исходное значение	0	0	0	0	0	0	0	0

Рис. 36. Регистр управления таймером/счетчиком 2 – **TCCR2**

Таблица 37

Выбор источника тактового сигнала таймера/счетчика 2

CS22	CS21	CS20	Описание
0	**0**	**0**	Таймер/счетчик 2 остановлен
0	**0**	**1**	PCK2
0	**1**	**0**	PCK2 / 8
0	**1**	**1**	PCK2 / 32
1	**0**	**0**	PCK2 / 64
1	**0**	**1**	PCK2 / 128
1	**1**	**0**	PCK2 / 256
1	**1**	**1**	PCK2 / 1024

Работа в режиме таймера. Таймер – устройство, обеспечивающее формирование сигнала по истечении заданного интервала времени.

Для обеспечения работы таймера необходимо иметь импульсы известной частоты и устройство, обеспечивающее формирование сигнала после поступления заданного количества импульсов.

В самом простом ТС0 из общей емкости счетчика $N = 2^n - 1$ вычитают заданное количество импульсов M, а разность записывают в ТС0 (**TCNT0**). Тогда после поступления M импульсов происходит переполнение счетчика и формируется сигнал переполнения.

Структура организации сравнения в таймере/счетчике 2 показана на рис. 37. ТС2 имеет встроенный компаратор, а в пространстве ввода/вывода выделен регистр сравнения **OCR2** с возможностью чтения и записи.

При совпадении содержимого счетчика (**TCNT0**) и регистра сравнения (**OCR2**) формируется сигнал прерывания по совпадению, а на выходе порта **PD7** устанавливается сигнал **OC2**. Вид этого сигнала определяется значениями битов **COM21** и **COM20** регистра управления **TCCR2** (табл. 38). Блок-схема таймера/счетчика 2 показана на рис. 38.

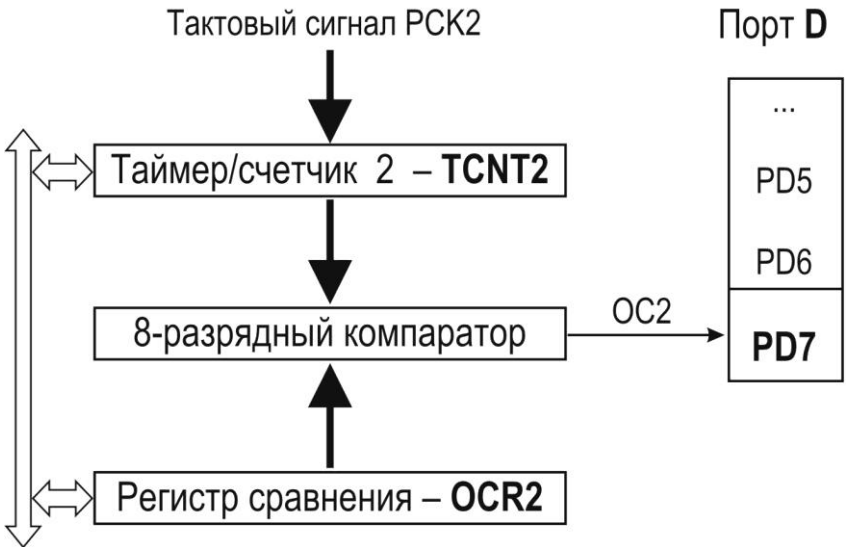


Рис. 37. Организация сравнения в таймере/счетчике 2

Таблица 38

Выбор режима сравнения

COM21	COM20	Описание
0	**0**	Таймер/счетчик отсоединен от выходного вывода **OC2**
0	**1**	Переключение выходной линии **OC2**
1	**0**	Очистка выходной линии **OC2** (установка в состояние 0)
1	**1**	Установка выходной линии **OC2** (установка в состояние 1)

Обратите внимание: В ШИМ-режиме функции этих битов отличаются.

TC1 имеет два встроенных компаратора – A и B, а в пространстве ввода/вывода выделено два 16-разрядных регистра сравнения **OCR1A** и **OCR1B** с возможностью чтения и записи. Порт D имеет, соответственно, два вывода сравнения **PD4** (**OC1B**) и **PD5** (**OC1A**). Вид сигнала на каждом из этих выводов определяется отдельно друг от друга битами **COM1A1**, **COM1A0** и **COM1B1**, **COM1B0** регистра управления **TCCR1A**.

Кроме этого, у ТС1 имеется функция захвата входа. То есть при поступлении внешнего сигнала на вход порта **PD6** (**ICP**) счет прекращается, формируется сигнал прерывания по захвату входа, а содержимое счетчика записывается в регистр захвата таймера/счетчика 1 **ICR1**. Этот режим обычно используется при работе аналогового компаратора.

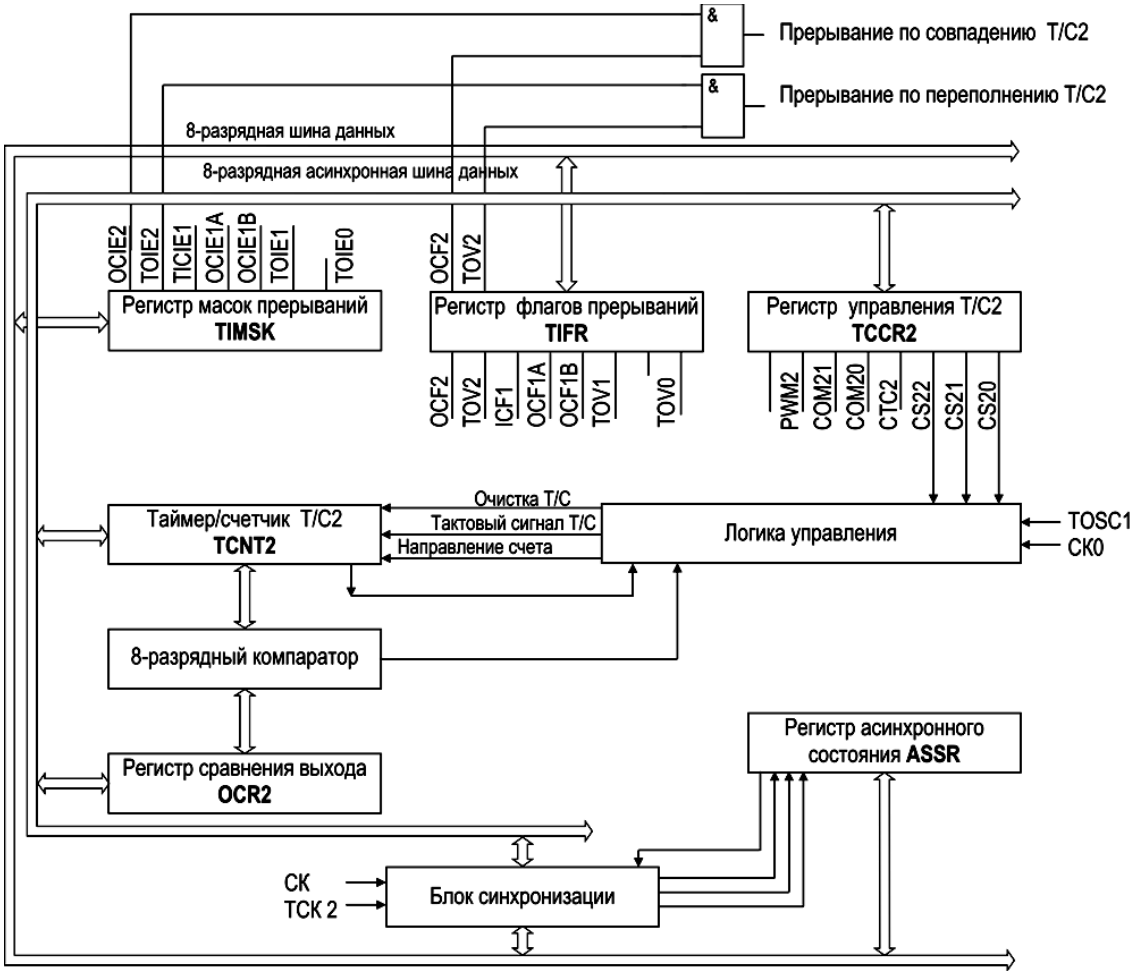


Рис. 38. Блок-схема таймера/счетчика 2

Работа в режиме широтно-импульсного модулятора. Таймеры/счетчики ТС1 и ТС2 могут работать в режиме широтно-импульсного модулятора.

Широтно-импульсный модулятор – это устройство, формирующее последовательность импульсов постоянной частоты, длительность которых пропорциональна величине входного сигнала.

Рассмотрим работу в режиме ШИМ на примере таймера/счетчика 2. При установленном ШИМ-режиме таймер/счетчик 2 и регистр сравнения выхода **OCR2** формируют 8-разрядный ШИМ-сигнал с выходом через выводы **PD7** (**OC2**).

Таймер/счетчик 2 работает как реверсивный счетчик, считающий от $00 до $FF, после чего он считает в обратную сторону до нуля и только после этого начинает новый цикл (рис. 39). В момент совпадения состояния счетчика с содержимым регистра сравнения выхода, на выводах **PD7** (**OC2**) устанавливается сигнал высокого или низкого уровня, в зависимости от комбинации значений **COM21**/**COM20** регистра управления **TCCR2.**

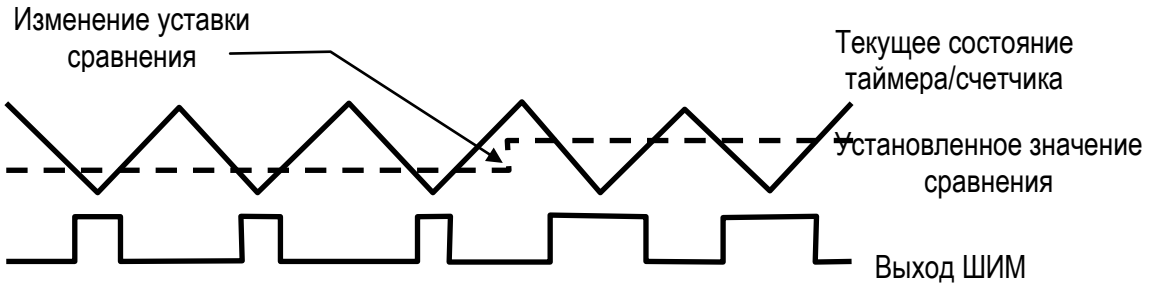


Рис. 39. Формирование ШИМ-сигнала

Частота ШИМ будет соответствовать тактовой частоте таймера, деленной на 512. Тип организации ШИМ определяется значениями битов **COM21** и **COM20** так, как показано в табл. 39.

Таблица 39

Выбор режима сравнения в ШИМ-режиме

C0M21	COM20	Эффект, оказываемый на вывод Compare/PWM
0	**0**	Не подсоединен
0	**1**	Не подсоединен
1	**0**	Очистка при совпадении, счет по нарастанию. Установка при совпадении, счет по убыванию (неинвертирующий ШИМ)
1	**1**	Очистка при совпадении, счет по убыванию. Установка при совпадении, счет по нарастанию (инвертирующий ШИМ)

В ШИМ-режиме флаг переполнения таймера **TOV2** устанавливается при смене направления счета при $00. Прерывание по переполнению таймера 2 работает так же, как и в нормальном режиме таймеров/счетчиков.

В табл. 40 приведены векторы прерывания, относящиеся к таймерам/счетчикам.

Таблица 40

Векторы прерывания таймеров/счетчиков

Vector No.	Program Address	Source	Interrupt Definition
4	$003	TIMER2 COMP	Совпадение при сравнении таймера/счетчика 2 (Timer/Conter2 Compare Match)
5	$004	TIMER2 OVF	Переполнение таймера/счетчика 2 (Timer/Conter2 Overflow)
6	$005	TIMER1 CAPT	Захват таймера/счетчика 1 (Timer/Conter1 Capture Event)
7	$006	TIMER1 COMPA	Совпадение A при сравнении таймера/счетчика 1 (Timer/Conter1 Compare Match A)
8	$007	TIMER1 COMPB	Совпадение B при сравнении таймера/счетчика 1 (Timer/Conter1 Compare Match B)
9	$008	TIMER1 OVF	Переполнение таймера/счетчика 1 (Timer/Conter1 Overflow)
10	$009	TIMER0 OVF	Переполнение таймера/счетчика 0 (Timer/Conter0 Overflow)

Варианты индивидуальных заданий

1. Запрограммируйте таймер/счетчик 0 для обеспечения индикации секунд на семисегментном индикаторе HG1. То есть каждую секунду на индикаторе должна меняться цифра 0, 1, 2, …, 9, 0, 1, 2, …, 9 и т.д. Частота кварцевого резонатора микроконтроллера – 8 МГц.

2. Запрограммируйте таймер/счетчик 0 для обеспечения динамической индикации цифр на семисегментных индикаторах HG1 и HG2. Таймер/счетчик 1 запрограммируйте для счета нажатий кнопки SB5. На индикацию программа должна выводить в виде шестнадцатеричного числа содержимое регистра TCNT1. При нажатии кнопки SB2 содержимое счетчика сбрасывается в нуль.

3. Запрограммируйте «бегущий» огонь на индикаторах VD4 – VD6, используя таймер/счетчик 0. Переключение тумблера SA1 должно изменять темп «бега». Для возможности визуального наблюдения «бега» огня придется реализовать программным путем счетчик числа формирований флага переполнения таймера/счетчика.

4. Запрограммируйте таймер/счетчик 1 для работы в ШИМ-режиме для выдачи импульсов на вывод **PD5** (**OC1A**) микроконтроллера, к которому подключен светодиод VD5. При включении тумблера SA1 частота ШИМ должна уменьшаться, а при включении SA2 – увеличиваться. Уменьшение и увеличение частоты импульсов оценивается по свечению светодиода VD5.

5. Запрограммируйте таймер/счетчик 2 для работы в ШИМ-режиме для выдачи импульсов на вывод **PD7** (**OC2**) микроконтроллера, к которому подключен звукоизлучатель HA1. При нажатии на кнопку SB1 частота ШИМ должна уменьшаться, а при одновременном нажатии двух кнопок SB1 и SB2 – увеличиваться. Уменьшение и увеличение частоты импульсов оценивается по изменению тона звучания звукоизлучателя.

6. Запрограммируйте таймер/счетчик 1 для реализации меандра (ступеней). Программа должна выводить импульсы меандра на звукоизлучатель HA1. Запрограммируйте циклическую программу линейно меняющейся частоты меандра (10 ступеней). После достижения максимальной частоты должен происходить ступенчатый переход на минимальную частоту.

7. На основе таймеров организуйте измерение длительности включенного состояния тумблера SA4 (**INT0**) с дискретностью 1 с. Допустимая длительность – 9 с. Индикацию секунд организуйте на семисегментном индикаторе HG2. При нажатии кнопки SB3 должно происходить обнуление индикатора.

8. Таймер/счетчик 0 и таймер/счетчик 1 запрограммируйте на счет внешних событий (нажатий и отпусканий кнопок SB4 и SB5 соответственно). На индикаторе HG2 программа должна индицировать разность содержимого счетчиков. Если число в таймере/счетчике 0 меньше, чем в таймере/счетчике 1, то на индикаторе должна высвечиваться цифра 0. Если число в таймере/счетчике 0 превышает больше чем на 9 число в таймере/счетчике 1, должна высвечиваться цифра 9. В остальных случаях высвечивается цифра превышения числа в таймере/счетчике 0 над числом в таймере/счетчике 1. При нажатии кнопки SB3 счетчики должны сбрасываться (обнуляться).

Контрольные вопросы

1. Поясните алгоритм функционирования таймеров микроконтроллеров AT90S8535.
2. Поясните алгоритм функционирования счетчиков микроконтроллеров AT90S8535.
3. Поясните алгоритм программирования микроконтроллеров.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Гребнев В.В. Микроконтроллеры семейства AVR фирмы Atmel / В.В. Гребнев. – М.: ИП Радиософт, 2002. – 176 с.

2. Зубков С.В. Assembler для DOS, Windows UNIX / С.В. Зубков. – М.: ДМК Пресс, 2000. – 608 с.

3. Козаченко В.Ф. Микроконтроллеры: руководство по применению 16-разрядных микроконтроллеров Intel MCS-196/296 во встроенных системах управления / В.Ф. Козаченко. – М.: ЭКОМ, 1997. – 688 с.

4. Лабораторный комплекс «Микроконтроллеры и автоматизация»: Техническое описание и методические указания к проведению лабораторных работ: ПМА90.00.000. – Челябинск: Уральский филиал РНПО «Росучприбор», 2005. – 138 с.

5. Лабораторный комплекс «Микроконтроллеры и автоматизация»: учеб. пособие. – Челябинск: Уральский филиал РНПО «Росучприбор», 2005. – 96 с.

6. Новиков Ю.В., Скоробогатов П.К. Основы микропроцессорной техники: Курс лекций: учеб. пособие / Ю.В. Новиков, П.К. Скоробогатов. – Изд-е 2-е, исправл. – М.: ИНТУИТ.РУ «Интернет-университет информационных технологий», 2004. – 440 с.

7. Трамперт В. AVR-RISC микроконтроллеры / В. Трамперт; пер. с нем. – Киев: МК-Пресс, 2006. – 464 с.

8. Финогенов К.Г. Использование языка Ассемблера: учеб. пособие для вузов / К.Г. Финогенов. – М.: Горячая линия – Телеком, 2004. – 438 с.

9. Кривченко И., Ламберт Е. AVR-микроконтроллеры: семь ярких лет становления. Что дальше? : [Электрон. ресурс] / И. Кривченко, Е. Ламберт. – Ч. 1. – URL: http://www.atmel.ru/Articles/Atmel25.htm (дата обращения 01.12.2008).

Приложение

ПРИМЕР ОФОРМЛЕНИЯ ОТЧЕТА ПО КУРСОВОМУ ПРОЕКТУ

КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ

Микропроцессорные средства в мехатронике и робототехнике

ЧАСТЬ 1. ИЗУЧЕНИЕ СИСТЕМЫ КОМАНД МИКРОКОНТРОЛЛЕРА И ДИРЕКТИВ АССЕМБЛЕРА

Группа	Выполнили	Оценка	Балл	Подпись
ЭАБ 1 -06	Толковкин Т.Т.			
Вариант **1**	Иванов Б.Б.			

Цель работы: ознакомиться с лабораторным комплексом «Микроконтроллеры и автоматизация», получить навык работы с программой AVR Studio, ознакомиться с системой команд программируемого микроконтроллера AT90S8535, подготовить простейшую программу, отладить ее, записать в микроконтроллер и продемонстрировать работу подготовленной программы.

Условие задачи: составьте программу вычитания из числа 5 числа 3. Если на блоке управления включен тумблер SA1, то на индикацию должен выдаваться результат вычитания. Если тумблер SA1 отключен, на индикацию выводится цифра ноль.

СТРУКТУРА ДАННЫХ

Таблица 1

Порядок настройки портов

Порт	Выводы на вход	Выводы на выход
Порт A	0–7	–
Порт B	0, 1, 4–7	2, 3
Порт C	–	0–7
Порт D	0–3	4–7

Таблица 2

Размещение структурных элементов программы

Элемент структуры данных	Имя	Адрес	Память
Вершина стека	SP	$025f	Регистры I/O
Начало программы	–	$0030	FLASH

Таблица 3

Перечень констант

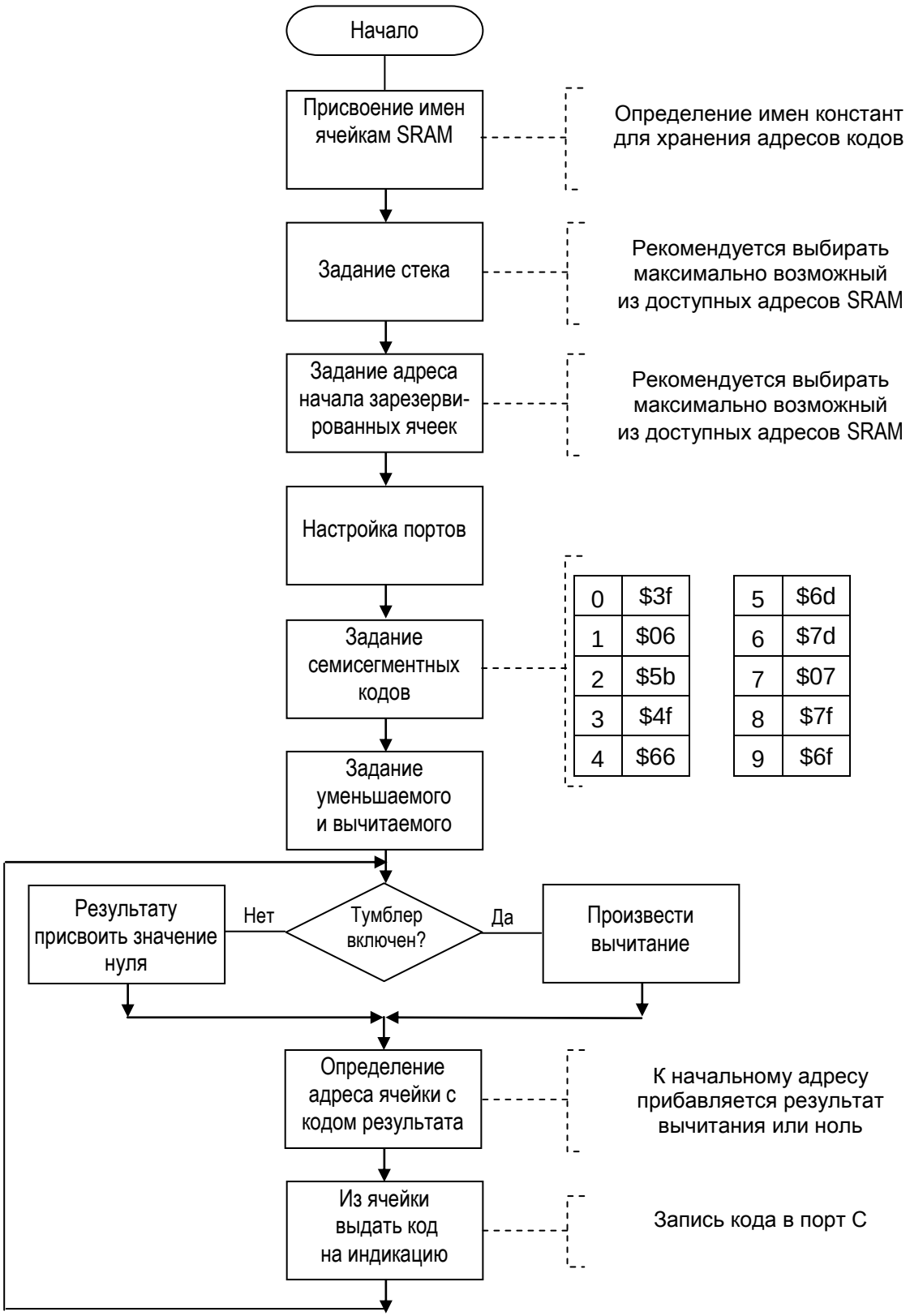
Константы	Имя	Значение	Адрес	Память
Адрес кода 0	cod0	$3f	$64	SRAM
Адрес кода 1	cod1	$06	$65	SRAM
Адрес кода 2	cod2	$5b	$66	SRAM
Адрес кода 3	cod3	$4f	$67	SRAM
Адрес кода 4	cod4	$66	$68	SRAM
Адрес кода 5	cod5	$6d	$69	SRAM
Адрес кода 6	cod6	$7d	$6a	SRAM
Адрес кода 7	cod7	$07	$6b	SRAM
Адрес кода 8	cod8	$7f	$6c	SRAM
Адрес кода 9	cod9	$6f	$6d	SRAM

Таблица 4

Перечень переменных

Переменные	Имя	Значение	Адрес	Память
Уменьшаемое	–	5	r17	РОН
Вычитаемое	–	3	r19	РОН
Результат (разность)	–	–	r20	РОН
Адрес ячейки кода индикации разности	–	–	r0	РОН
Адрес первого из кодов индикации	–	$64	Z (zh, zl) (r31, r30)	РОН
Вспомогательные регистры для промежуточных значений	–	–	r16, r17	РОН

АЛГОРИТМ


Начало
Присвоение имен ячейкам SRAM
Определение имен констант для хранения адресов кодов
Задание стека
Рекомендуется выбирать максимально возможный из доступных адресов SRAM
Задание адреса начала зарезерви-рованных ячеек
Рекомендуется выбирать максимально возможный из доступных адресов SRAM
Настройка портов
Задание семисегментных кодов
0 $3f
1 $06
2 $5b
3 $4f
4 $66
5 $6d
6 $7d
7 $07
8 $7f
9 $6f
Задание уменьшаемого и вычитаемого
Результату присвоить значение нуля
Нет
Тумблер включен?
Да
Произвести вычитание
Определение адреса ячейки с кодом результата
К начальному адресу прибавляется результат вычитания или ноль
Из ячейки выдать код на индикацию
Запись кода в порт С

ТЕКСТ ПРОГРАММЫ

```
;***************************************************
; Работа № 1. Вариант № 1
;************************************ **********************
;******подключаемые дополнительные файлы******
.include "8535def.inc"       ; включить файл – описание для AT90S8535

.dseg                        ; объявление сегмента данных

;******* ; присвоение имен ячейкам SRAM **********************
.equ   cod0=$64
.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                        ; объявление сегмента кодов
.org   0                     ; адрес начала программы в программной памяти
rjmp   reset                 ; прерывание по сбросу

;*******начало основной программы**********
.org   $30
reset:

; ******* задание стека с вершиной по адресу $025f *******
ldi     r16,$02
out     sph,r16
ldi     r16,$5f
out     spl,r16

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

```
        ; *******    Настройка портов     ****************************
        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 (активен HG2)

        ; *******   задание семисегментных кодов    *******
        ldi     r17,$3f
        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

        ;******* Задание уменьшаемого и вычитаемого    *******
        ldi     r17,5                   ; задание уменьшаемого
        ldi     r18,3                   ; задание вычитаемого

        m1:
```

```
;******* Проверка «Тумблер включен?»    *******
sbis pina,4          ; если включен тумблер SA1, то пропустить
rjmp   m2             ; следующую команду

;******* Произвести вычитание    *******
mov   r20,r17        ; в r20 поместить уменьшаемое
sub    r20,r18        ; вычесть вычитаемое, результат остается в r20
rjmp   vv

;******* Результату присвоить значение нуля    *******

m2:
ldi      r20,0          ; в r20 записать ноль

;******* Определение  адреса ячейки с кодом результата    *******

vv:

push   zl                    ; сохранить zl в стеке
add    zl,r20                ; сложить zl с результатом
ld       r0,z                  ;семисегментный код результата переслать в r0
pop     zl                    ; извлечь zl из стека

;******* Из ячейки  выдать код  на индикацию    *******
out      portc,r0              ; запись кода в порт С

rjmp   m1
```

ВЫВОДЫ ПО РАБОТЕ:

Выполнили _____ Толковкин Т.Т.
(Подпись)
_____ Иванов Б.Б.
(Подпись)

_____
(Подпись)

СОДЕРЖАНИЕ

Учебное издание

Микропроцессорные средства в мехатронике и робототехнике

Лабораторный практикум

Козелков Олег Владимирович,
Ломакин Игорь Владимирович

Кафедра приборостроения и мехатроники КГЭУ

Редактор издательского отдела *Н.А. Мустакимова*
Компьютерная верстка *Т.И. Лунченкова*

Подписано в печать 11,05.2020.
Формат 60×84/16. Бумага ВХИ. Гарнитура «Times». Вид печати РОМ.
Усл. печ. л. 5,52. Уч.-изд. л. 6,12. Тираж 500 экз. Заказ 122/эл.

Редакционно-издательский отдел КГЭУ,
420066, Казань, Красносельская, 51