Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Мордовский государственный университет им. Н. П. Огарева»

А.Ю.Бальзамов

МИКРОКОНТРОЛЛЕРЫ AVR: АРХИТЕКТУРА И ПРОГРАММИРОВАНИЕ

Электронное учебное пособие

Саранск
2014

УДК [004.42:621.316.544.1](075.8)

Рецензенты:

Карасев А. В., кандидат технических наук, доцент кафедры электроники и электротехники:

Сурин Б. П., кандидат физико-математических наук, старший научный сотрудник ЗАО НПК «Электровыпрямитель».

Бальзамов А. Ю. Микроконтроллеры AVR: архитектура и программирование: учебное пособие / А.Ю.Бальзамов; Мордов. гос. ун-т. – Саранск, 2014. – 2,6 Мб

Об авторе: Бальзамов А. Ю. – кандидат технических наук, доцент кафедры электроники и наноэлектроники ФГБОУ ВПО «МГУ им. Н. П. Огарёва»

В учебном пособии рассматривается архитектура популярных микроконтроллеров семейства AVR фирмы «Atmel», система команд и ассемблер этих микроконтроллеров. Приводится описание лабораторного комплекса «Микроконтроллеры и автоматизация» на базе микроконтроллеров ATmega8535 и интегрированной среды разработки программ AVR-Studio. Предлагаются варианты лабораторных работ для изучения функционирования AVR-микроконтроллеров.

Пособие предназначено для организации лабораторных занятий по курсам «Основы микропроцессорной техники», «Микропроцессорные средства» и т. п., изучаемым студентами направлений «Электроника и наноэлектроника», «Информатика и вычислительная техника» и других инженерных направлений и специальностей.

СОДЕРЖАНИЕ

Предисловие

Целью настоящего электронного учебного пособия является развитие навыков по программированию микроконтроллерных систем на языке ассемблера, ознакомление с аппаратными средствами таких систем, программными средствами проектирования и отладки. Студент, приступающий к изучению изложенного в пособии материала, должен владеть основами информатики и цифровой электронной техники.

Микроконтроллерные системы изучаются на примере одних из наиболее популярных в настоящее время в мире микроконтроллеров AVR фирмы «Atmel». Для обучения используется лабораторный комплекс «Микроконтроллеры и автоматизация» разработки научно-производственного предприятия «Учебная техника – Профи» (г. Челябинск), а также интегрированная отладочная среда разработки программ AVR-Studio. Дается краткое описание архитектуры и системы команд AVR-микроконтроллеров, ассемблерных директив и особенностей построения ассемблерных программ, структуры лабораторного комплекса и основных возможностей AVR-Studio.

Лабораторные работы по программированию AVR-микроконтроллеров рассчитаны на освоение их основных программно-аппаратных средств: системы команд, портов ввода-вывода, системы прерываний, таймеров/счетчиков, аналого-цифрового преобразователя, а также особенностей построения микроконтроллерных систем автоматизации производства. В каждой лабораторной работе дается необходимый теоретический материал и предлагается 16 различных вариантов индивидуальных заданий. Для облегчения процесса написания студентами индивидуальных программ приводится ряд примеров составления аналогичных программ.

1. АРХИТЕКТУРА МИКРОКОНТРОЛЛЕРОВ AVR СЕМЕЙСТВА MEGA ФИРМЫ «ATMEL»

1.1. Общие сведения

Микроконтроллеры AVR появились на рынке в конце 1990-х годов и в короткие сроки завоевали высокую популярность. С каждым годом они захватывают все новые и новые ниши на рынке. Не последнюю роль в этом играет соотношение показателей цена/быстродействие/энергопотребление, до сих пор являющееся едва ли не лучшим на рынке 8-битных микроконтроллеров. Кроме того, постоянно растет число выпускаемых сторонними производителями разнообразных программных и аппаратных средств поддержки разработок устройств на их основе. Все это позволяет говорить о микроконтроллерах AVR как об индустриальном стандарте среди 8-битных микроконтроллеров.

В настоящее время в рамках единой базовой архитектуры микроконтроллеры AVR подразделяются на несколько семейств, среди которых основными являются:

- Classic AVR (снимается с производства);
- Tiny AVR;
- Mega AVR;
- Mega AVR для специальных применений.

Микроконтроллеры семейства Tiny имеют небольшие объемы памяти программ и данных и весьма ограниченную периферию. Практически все они выпускаются в 8-выводных корпусах и предназначены для интеллектуальных датчиков различного назначения (контрольные, охранные, пожарные), игрушек, зарядных устройств, различной бытовой техники.

Микроконтроллеры семейства Mega имеют наиболее развитую периферию, наибольшие среди всех микроконтроллеров AVR объемы памяти программ и данных. Они предназначены для использования в мобильных телефонах, в контроллерах различного периферийного оборудования (такого как принтеры, сканеры, современные дисковые накопители, приводы CD-ROM/DVD-ROM и т. п.), в сложной офисной технике и т. д.

Процессорное ядро (CPU) AVR-микроконтроллеров (рис. 1) выполнено по усовершенствованной RISC-архитектуре (enhanced RISC), в которой используется ряд решений, направленных на повышение быстродействия микроконтроллеров.

Арифметико-логическое устройство (АЛУ), выполняющее все вычисления, подключено непосредственно к 32 рабочим регистрам, объединенным в регистровый файл. Благодаря этому АЛУ может выполнять одну операцию (чтение содержимого регистров, выполнение операции и запись результата обратно в регистровый файл) за такт. Кроме того, практически каждая из команд (за исключением команд, у которых одним из операндов является 16-битный адрес) занимает одну ячейку памяти программ.

В микроконтроллерах AVR реализована Гарвардская архитектура, характеризующаяся раздельной памятью программ и данных, каждая из которых

имеет собственные шины доступа. Такая организация позволяет одновременно работать как с памятью программ, так и с памятью данных.

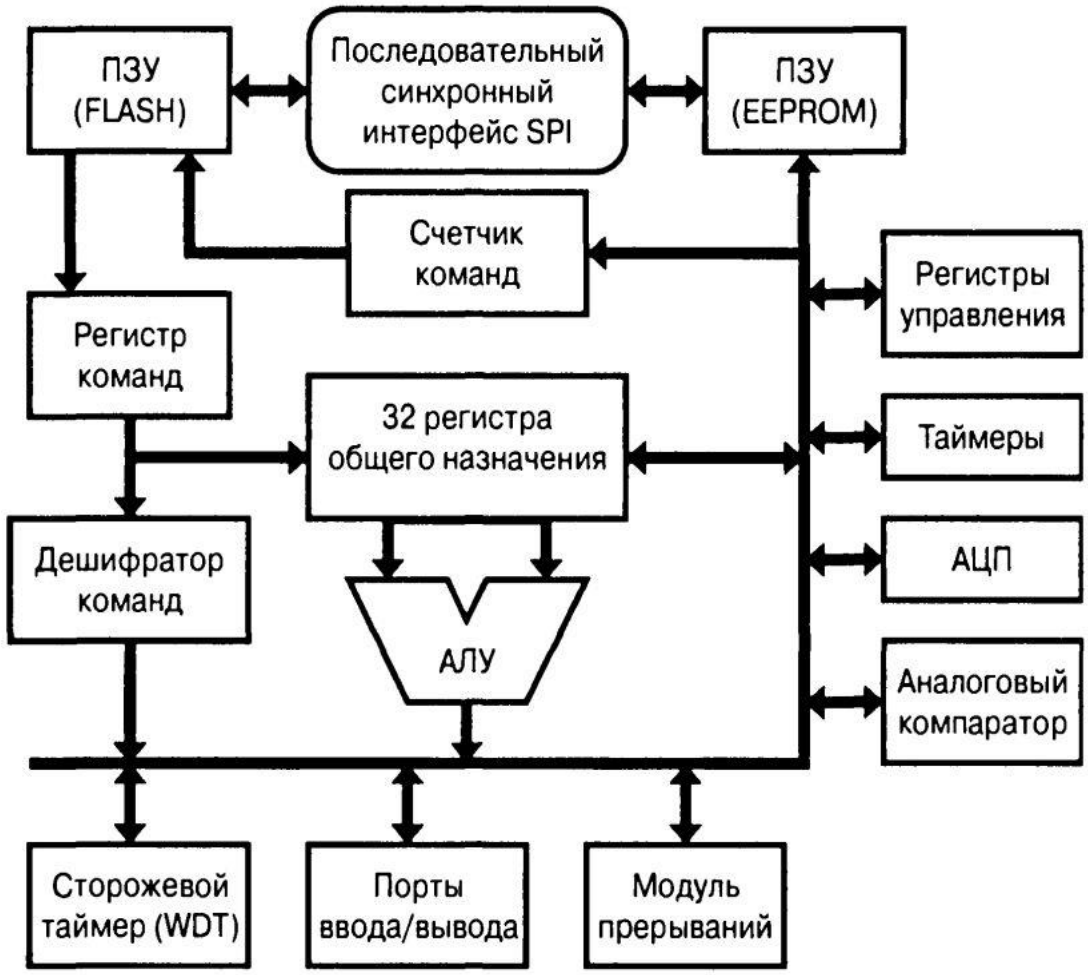


Рис.1. Архитектура процессорного ядра AVR-микроконтроллера

Разделение информационных шин позволяет использовать для каждого типа памяти шины различной разрядности, причем способы адресации и доступа к каждому типу памяти также различаются. В сочетании с двухуровневым конвейером команд (одна команда выполняется, другая параллельно выбирается и декодирутся) такая архитектура позволяет достичь производительности в 1 MIPS на 1 МГц тактовой частоты.

AVR-микроконтроллеры изготавливаются по малопотребляющей КМОП-технологии и имеют полностью статическую архитектуру, минимальная тактовая частота равна нулю.

Основные параметры и характеристики микроконтроллеров AVR семейства Mega:

• FLASH-память программ объемом от 8 до 256 Кбайт (число циклов стирания/записи не менее 10 000);

• оперативная память (статическое ОЗУ) объемом от 512 байт до 8 Кбайт;

• память данных на основе ЭСППЗУ (EEPROM) объемом от 256 байт до 4 Кбайт (число циклов стирания/записи не менее 100 000);

• возможность защиты от чтения и модификации памяти программ и данных;

• возможность программирования непосредственно в системе через последовательные интерфейсы SPI и JTAG;

• возможность самопрограммирования;

• возможность внутрисхемной отладки в соответствии со стандартом IEEE 1149.1 (JTAG), а также наличие собственного однопроводного интерфейса внутрисхемной отладки (debugWire1);

• разнообразные способы синхронизации: встроенный RC-генератор с внутренней или внешней времязадающей RC-цепочкой, встроенный генератор с внешним кварцевым или пьезокерамическим резонатором, внешний сигнал синхронизации;

• наличие нескольких режимов пониженного энергопотребления;

• наличие сторожевого таймера;

• наличие детектора пониженного напряжения питания (Brown-Out Detector-BOD);

• возможность программного снижения частоты тактового генератора.

• векторная система прерываний, поддержка очереди прерываний;

• большое число источников прерываний (до 45 внутренних и до 32 внешних);

• наличие аппаратного умножителя.

1.2. Периферийные модули

Микроконтроллеры AVR семейства Mega имеют от 23 до 86 линий ввода/вывода, которые объединяются в 8-разрядные порты ввода/вывода, причем отдельные линии могут быть запрограммированы как входные или как выходные независимо друг от друга. Каждая линия имеет входной буфер с триггером Шмита а индивидуально отключаемый внутренний подтягивающий резистор сопротивлением 20...50 кОм.

Кроме того, имеется богатый набор периферийных устройств:

• один или два 8-битных таймера/счетчика, во всех моделях с двумя 8-битными таймерами/счетчиками один из них может работать в качестве часов реального времени (в асинхронном режиме);

• от одного до четырех 16-битных таймеров/счетчиков;

• одно- и двухканальные генераторы 8-битного ШИМ-сигнала (один из режимов работы 8-битных таймеров/счетчиков);

• двух- и трехканальные генераторы ШИМ-сигнала регулируемой разрядности (один из режимов работы 16-битных таймеров/счетчиков), разрешение формируемого сигнала может составлять от 1 до 16 бит;

• аналоговый компаратор;

• многоканальный 10-битный АЦП последовательного приближения, имеющий как несимметричные, так и дифференциальные входы;

• последовательный синхронный интерфейс SPI;

• последовательный двухпроводный интерфейс TWI (полный аналог интерфейса I^2C);

• от одного до четырех полнодуплексных универсальных синхронных/асинхронных приемо-передатчиков (USART), которые в ряде моделей могут использоваться в качестве ведущего устройства шины SPI;

• универсальный последовательный интерфейс USI, который может использоваться в качестве интерфейса SPI или I^2C, а также полудуплексного UART или 4/12-битного счетчика.

В качестве примера на рис. 2 приведена структурная схема микроконтроллера ATmega8535, который имеет четыре 8-битных порта ввода/вывода (порты A...D), два 8-битных (T0, T2) и один 16-битный (T1) таймер/счетчик, 4 канала ШИМ, по одному интерфейсному модулю USART, SPI и TWI.

1.3. Конструктивное исполнение

В семейство Mega на сегодняшний день входит в общей сложности 24 модели микроконтроллеров, которые делятся на 4 группы.

1. Микроконтроллеры в 32-выводных корпусах типа TQFP и MLF (также выпускаются в 28-выводных корпусах типа DIP) с максимальным числом контактов ввода/вывода, равным 23.

2. Микроконтроллеры в 44-выводных корпусах типа TQFP и MLF (также выпускаются в 40-выводных корпусах типа DIP) с максимальным числом контактов ввода/вывода, равным 35 (модели с возможностью подключения внешнего ОЗУ) или 32 (остальные модели).

3. Микроконтроллеры в 64-выводных корпусах типа TQFP и MLF, имеют 53 или 54 контакта ввода/вывода.

4. Микроконтроллеры в 100-выводных корпусах типа TQFP, имеют 68 или 86 контактов ввода/вывода.

Некоторые из моделей выпускаются взамен снятых с производства микроконтроллеров семейства Classic, по цоколевке и функционально совместимы с ними, например, микроконтроллер ATmega8535 выпускается взамен AT90S8535.

Следует отметить, что одни модели микроконтроллеров семейства выпускаются как в коммерческом (диапазон рабочих температур 0 … +70 °C), так и в промышленном (диапазон рабочих температур —40 ... +85 °C) исполнениях, а другие — только в промышленном. Напряжение питания в различных исполнениях составляет 4,5 … 5,5, 2,7 … 5,5 или 1,8 … 5,5 В. Максимальная тактовая частота может составлять 4, 8, 10, 16 или 20 МГц.

1.4. Организация памяти

Обобщенная карта памяти микроконтроллеров AVR семейства Mega приведена на рис. 3. Поскольку микроконтроллеры AVR имеют 16-битную систему команд, объем памяти программ на рисунке указан не в байтах, а в 16-битных словах. Символ «$» перед числом означает, что это число записано в шестнадцатеричной системе счисления. Для микроконтроллера ATmega8535 объем памяти программ составляет 8 Кбайт (4К*16), ее верхняя граница F_END = $FFF, объем ОЗУ данных – 512 байт, верхняя граница S_END = $25F, объем ПЗУ данных – также 512 байт, верхняя граница E_END = $1FF.

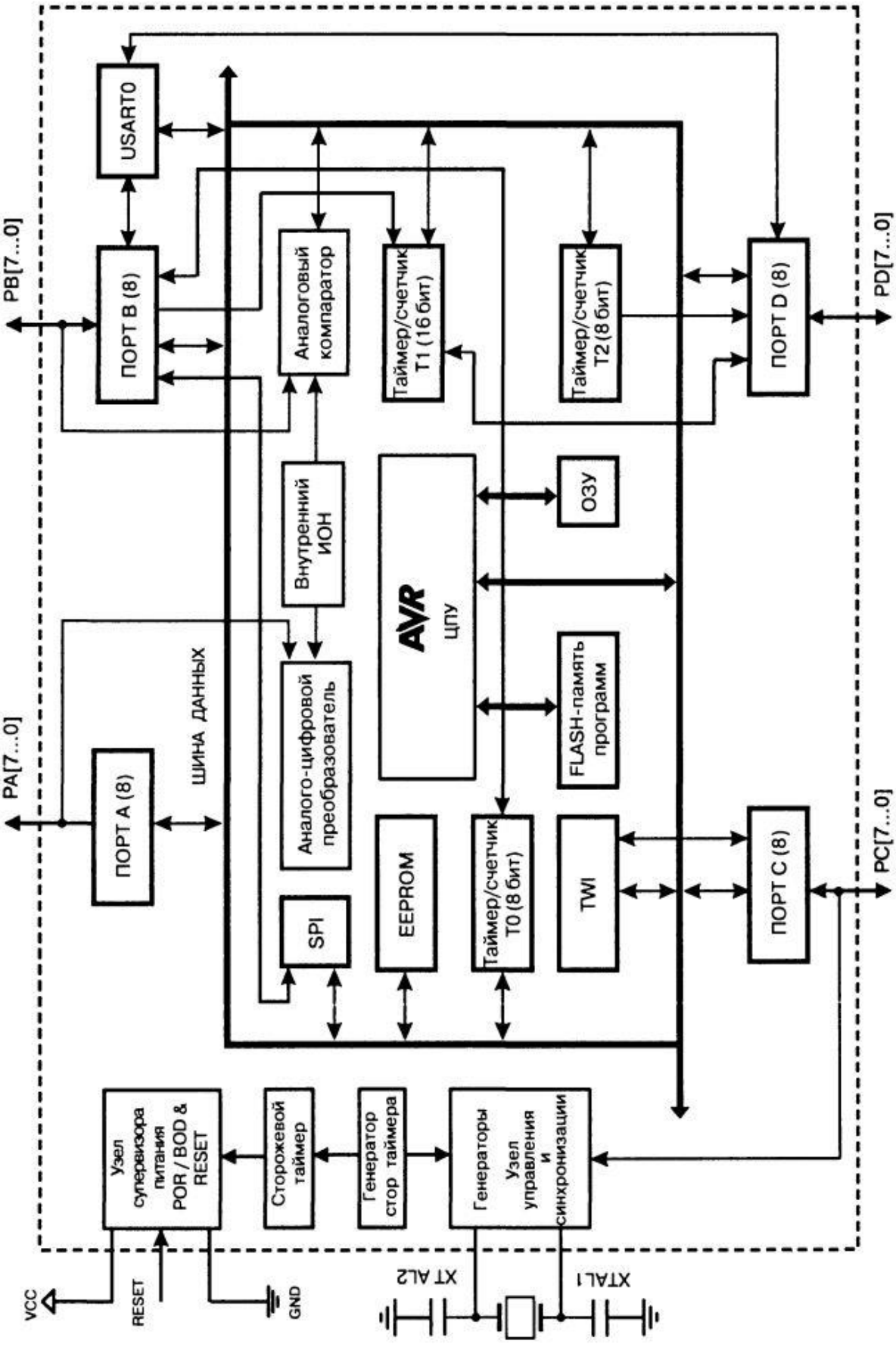


Рис. 2. Структурная схема микроконтроллера ATmega8535

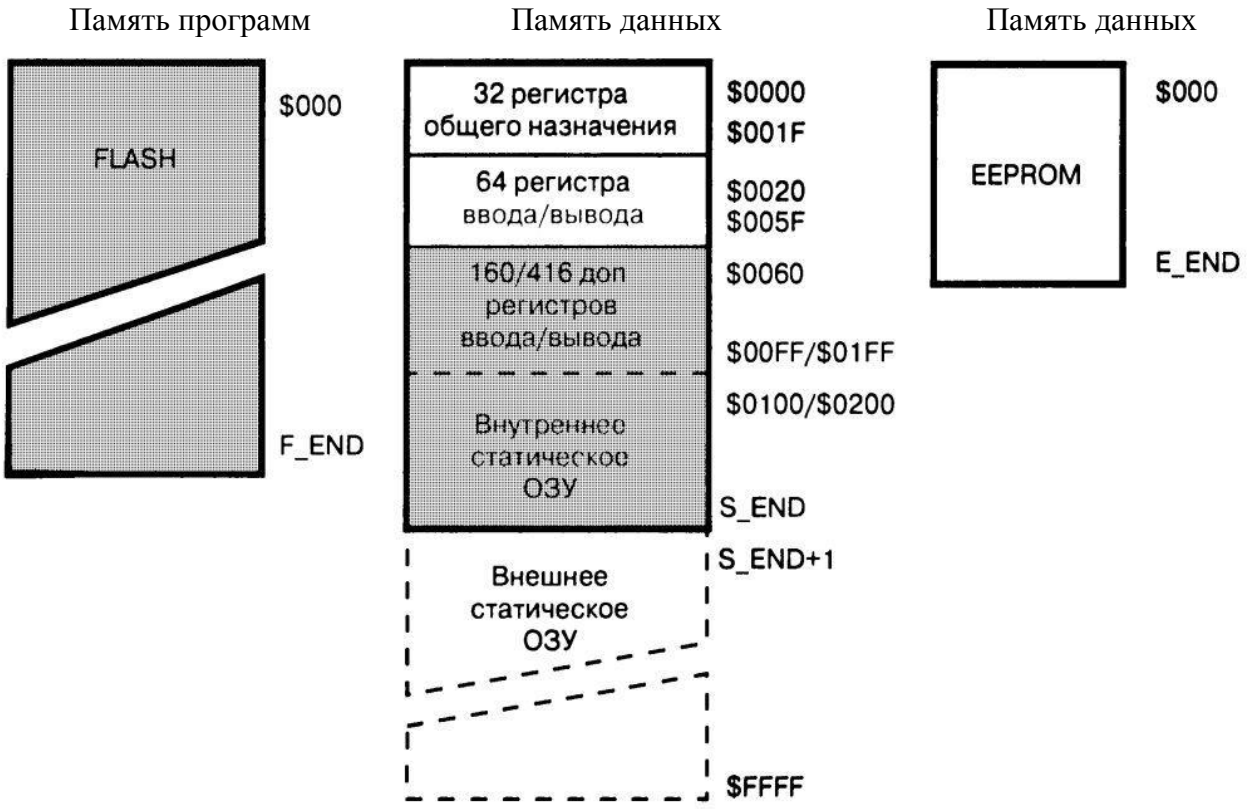


Рис. 3. Карта памяти AVR-микроконтроллеров семейства Mega

Память программ (объемом от 8 до 256 Кбайт) предназначена для хранения команд, управляющих работой микроконтроллера, а также часто используется для хранения таблиц констант, не меняющихся во время работы программы. Для пересылки байта из памяти программ в память данных существует специальная команда — LPM. При использовании команды LPM адрес, по которому производится чтение, определяется содержимым индексного регистра Z. При этом старшие 15 битов содержимого регистра будут определять адрес слова (0 ... 32 К), младший бит будет определять, какой из байтов будет прочитан: 0 — младший байт, 1 — старший байт.

В подавляющем большинстве моделей микроконтроллеров семейства Mega память программ логически разделена на две неравные части: область прикладной программы и область загрузчика. В последней может располагаться специальная программа (загрузчик), позволяющая микроконтроллеру самостоятельно управлять загрузкой и выгрузкой прикладных программ. Если же возможность самопрограммирования микроконтроллера не используется, прикладная программа может располагаться и в области загрузчика.

По адресу $0000 памяти программ находится вектор сброса. После инициализации (сброса) микроконтроллера выполнение программы начинается с этого адреса (по этому адресу должна размещаться команда перехода к инициализационной части программы). Начиная с адреса $001 (модели с памятью программ 8 Кбайт и меньше) или $0002 (остальные модели) памяти программ располагается таблица векторов прерываний. Размер этой области зависит от модели микроконтроллера.

При возникновении прерывания после сохранения в стеке текущего значения счетчика команд происходит выполнение команды, расположенной по адресу соответствующего вектора. Поэтому по данным адресам располагаются команды перехода к подпрограммам обработки прерываний. В моделях с памятью программ небольшого объема (8 Кбайт и менее) в таблице векторов прерываний используются команды относительного перехода (RJMP), а в остальных моделях — команды абсолютного перехода (JMP). Если в программе прерывания не используются, то основная программа может начинаться непосредственно с адреса $0001.

Память данных микроконтроллеров семейства Mega разделена на три части: регистровая память, оперативная память (статическое ОЗУ) и энергонезависимое ЭСППЗУ (EEPROM).

Регистровая память включает 32 регистра общего назначения (РОН), и 64 служебных регистра ввода/вывода (РВВ). В сложных моделях с развитой периферией имеется также область дополнительных (extended) регистров ввода/вывода (ДРВВ). Под РВВ в памяти микроконтроллера отводится 64 байта, а под ДРВВ — 160 или 416 байт (в зависимости от модели).

В отличие от процессоров с аккумулятором, в процессорном ядре AVR все 32 РОН непосредственно доступны АЛУ. Благодаря этому любой РОН может использоваться практически во всех командах и как операнд-источник, и как операнд-приемник. Последние 6 регистров общего назначения (R26...R31) могут также объединяться в три 16-битных регистра X, Y и Z, используемых в качестве указателей при косвенной адресации памяти данных.

В областях регистров ввода/вывода располагаются различные служебные регистры (регистр управления микроконтроллера, регистр состояния и т. п.), а также регистры управления периферийными устройствами, входящими в состав микроконтроллера. Общее их количество зависит от конкретной модели микроконтроллера.

К регистрам ввода/вывода, расположенным в основном пространстве ввода/вывода, можно напрямую обратиться с помощью команд IN и OUT, выполняющих пересылку данных между одним из 32 РОН и пространством ввода/вывода. В системе команд имеется также четыре команды побитового доступа, использующие в качестве операндов регистры ввода/вывода: команды установки/сброса отдельного бита (SBI и CBI) и команды проверки состояния отдельного бита (SBIS и SBIC). Но эти команды могут обращаться только к 1-й половине основного пространства ввода/вывода (адреса $00...$1F).

Среди РВВ есть один регистр, используемый наиболее часто в процессе выполнения программ. Это регистр состояния SREG. Он располагается по адресу $3F ($5F) и содержит набор флагов, показывающих текущее состояние микроконтроллера. Большинство флагов автоматически устанавливаются в 1 или сбрасываются в 0 при наступлении определенных событий (в соответствии с результатом выполнения команд). Все биты этого регистра доступны как для чтения, так и для записи; после сброса микроконтроллера все биты регистра сбрасываются в 0. Формат этого регистра следующий:

Регистр состояния – SREG

Бит	7	6	5	4	3	2	1	0	
$3F ($5F)	I	T	H	S	V	N	Z	C	**SREG**
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	

• Бит 7 – **I: Общее разрешение прерываний**. Для разрешения прерываний этот флаг должен быть установлен в «1». Разрешение/запрещение отдельных прерываний производится установкой или сбросом соответствующих разрядов регистров масок прерываний (регистра управления прерываниями). Если флаг сброшен, то прерывания запрещены независимо от состояния разрядов этих регистров. Флаг сбрасывается аппаратно после входа в прерывание и восстанавливается командой RETI для разрешения обработки следующих прерываний.

• Бит 6 – **T: Хранение копируемого бита.** Этот разряд регистра используется в качестве источника или приемника команд копирования битов BLD (Bit LoaD) и BST (Bit Store). Заданный разряд любого РОН может быть скопирован в этот разряд командой BST или установлен в соответствии с содержимым данного разряда командой BLD.

• Бит 5 – **H: Флаг половинного переноса.** Этот флаг устанавливается в «1», если произошел перенос из младшей половины байта в старшую (из 3-го разряда в 4-й) или заем из старшей половины байта при выполнении некоторых арифметических операций.

• Бит 4 – **S: Флаг знака.** Этот флаг равен результату операции «Исключающее ИЛИ» между флагами N (отрицательный результат) и V (переполнение числа в дополнительном коде). Соответственно этот флаг устанавливается в «1», если результат выполнения арифметической операции меньше нуля.

• Бит 3 – **V: Флаг переполнения дополнительного кода.** Этот флаг устанавливается в «1», при переполнении разрядной сетки знакового результата. Используется при работе со знаковыми числами (представленными в дополнительном коде).

• Бит 2 – **N: Флаг отрицательного значения.** Этот флаг устанавливается в «1», если старший (7-й) разряд результата операции равен «1». В противном случае флаг равен «0».

• Бит 1 – **Z: Флаг нулевого значения.** Этот флаг устанавливается в «1», если результат выполнения операции равен нулю.

• Бит 0 – **C: Флаг переноса.** Этот флаг устанавливается в «1», если в результате выполнения операции произошел выход за границы байта.

В табл. 1 приведены в качестве примера состав, назначение и адреса всех регистров ввода-вывода микроконтроллера ATmega8535.

Для хранения переменных помимо регистров общего назначения также используется статическое ОЗУ объемом от 512 байт до 8 Кбайт. Ряд микроконтроллеров семейства, кроме того, имеют возможность подключения внешнего статического ОЗУ объемом до 64 Кбайт.

В адресном пространстве ОЗУ также расположены все регистры микроконтроллеров, под них отведены младшие 96 (256) адресов. Каждый регистр имеет свой собственный адрес в пространстве памяти данных. Поэтому к регистрам можно обращаться двумя способами — как к регистрам и как к памяти, несмотря на то, что физически эти регистры не являются ячейками ОЗУ. Так, регистрам общего назначения R0 – R31 соответствуют адреса ОЗУ $000 – $01F, регистрам ввода/вывода $00 – $3F соответствуют адреса ОЗУ $020 – $05F (номер регистра плюс $20).

Таблица 1. Регистры ввода/вывода микроконтроллера ATmega8535

Название	Адрес	Функция
SREG	$3F ($5F)	Регистр состояния
SPH	$3E ($5E)	Указатель стека, старший байт
SPL	$3D ($5D)	Указатель стека, младший байт
OCR0	$3C ($5C)	Регистр совпадения таймера/счетчика T0
GICR	$3B ($5B)	Общий регистр управления прерываниями
GIFR	$3A ($5A)	Общий регистр флагов прерываний
TIMSK	$39 ($59)	Регистр маски прерываний от таймеров/счетчиков
TIFR	$38 ($58)	Регистр флагов прерываний от таймеров/счетчиков
SPMCR	$37 ($57)	Регистр управления и состояния операций записи в память программ
TWCR	$36 ($56)	Регистр управления TWI
MCUCR	$35 ($55)	Регистр управления микроконтроллера
MCUCSR	$34 ($54)	Регистр управления и состояния микроконтроллера
TCCR0	$33 ($53)	Регистр управления таймера/счетчика T0
TCNT0	$32 ($52)	Счетный регистр таймера/счетчика T0
OSCCAL	$31 ($51)	Регистр калибровки тактового генератора
SFIOR	$30 ($50)	Регистр специальных функций
TCCR1A	$2F ($4F)	Регистр A управления таймера/счетчика T1
TCCR1B	$2E ($4E)	Регистр B управления таймера/счетчика T1
TCNT1H	$2D ($4D)	Счетный регистр таймера/счетчика T1, старший байт
TCNT1L	$2C ($4C)	Счетный регистр таймера/счетчика T1, младший байт
OCR1AH	$2B ($4B)	Регистр A совпадения таймера/счетчика T1, старший байт
OCR1AL	$2A ($4A)	Регистр A совпадения таймера/счетчика T1, младший байт
OCR1BH	$29 ($49)	Регистр B совпадения таймера/счетчика T1, старший байт
OCR1BL	$28 ($48)	Регистр B совпадения таймера/счетчика T1, младший байт
ICR1H	$27 ($47)	Регистр захвата таймера/счетчика T1, старший байт
ICR1L	$26 ($46)	Регистр захвата таймера/счетчика T1, младший байт
TCCR2	$25 ($45)	Регистр управления таймера/счетчика T2
TCNT2	$24 ($44)	Счетный регистр таймера/счетчика T2
OCR2	$23 ($43)	Регистр совпадения таймера/счетчика T2
ASSR	$22 ($42)	Регистр состояния асинхронного режима
WDTCR	$21 ($41)	Регистр управления сторожевого таймера
UBRRH	$20 ($40)	Регистр скорости передачи USART, старший байт
UCSRC		Регистр управления и состояния USART

Окончание табл. 1

Название	Адрес	Функция
EEARH	$1F ($3F)	Регистр адреса EEPROM, старший байт
EEARL	$1E ($3E)	Регистр адреса EEPROM, младший байт
EEDR	$1D ($3D)	Регистр данных EEPROM
EECR	$1C ($3C)	Регистр управления EEPROM
PORTA	$1B ($3B)	Регистр данных порта A
DDRA	$1A ($3A)	Регистр направления данных порта A
PINA	$19 ($39)	Выводы порта A
PORTB	$18 ($38)	Регистр данных порта B
DDRB	$17 ($37)	Регистр направления данных порта B
PINB	$16 ($36)	Выводы порта B
PORTC	$15 ($35)	Регистр данных порта C
DDRC	$14 ($34)	Регистр направления данных порта C
PINC	$13 ($33)	Выводы порта C
PORTD	$12 ($32)	Регистр данных порта D
DDRD	$11 ($31)	Регистр направления данных порта D
PIND	$10 ($30)	Выводы порта D
SPDR	$0F ($2F)	Регистр данных SPI
SPSR	$0E ($2E)	Регистр состояния SPI
SPCR	$0D ($2D)	Регистр управления SPI
UDR	$0C ($2C)	Регистр данных USART
UCSRA	$0B ($2B)	Регистр A управления и состояния USART
UCSRB	$0A ($2A)	Регистр B управления и состояния USART
UBRRL	$09 ($29)	Регистр скорости передачи USART, младший байт
ACSR	$08 ($28)	Регистр управления и состояния аналогового компаратора
ADMUX	$07 ($27)	Регистр управления мультиплексором АЦП
ADCSRA	$06 ($26)	Регистр A управления и состояния АЦП
ADCH	$05 ($25)	Регистр данных АЦП, старший байт
ADCL	$04 ($24)	Регистр данных АЦП, младший байт
TWDR	$03 ($23)	Регистр данных TWI
TWAR	$02 ($22)	Регистр адреса TWI
TWSR	$01 ($21)	Регистр состояния TWI
TWBR	$00 ($20)	Регистр скорости передачи TWI

В ОЗУ данных может быть организован стек, для использования которого необходимо проинициализировать указатель стека (пару регистров ввода/вывода SPH:SPL). Запись в стек выполняется в сторону уменьшения адресов памяти. В стек автоматически заносится адрес возврата при вызове подпрограммы или генерации прерывания, а также можно программно записать и считать любую информацию при помощи команд занесения в стек (PUSH) и извлечения из стека (POP).

Для долговременного хранения различной информации, которая может изменяться в процессе функционирования готовой системы (калибровочные

константы, серийные номера, ключи и т. п.), в микроконтроллерах семейства может использоваться встроенная EEPROM-память. Ее объем составляет для различных моделей от 256 байт до 4 Кбайт. Эта память расположена в отдельном адресном пространстве, а доступ к ней осуществляется с помощью трех регистров ввода/вывода: регистра адреса, регистра данных и регистра управления.

Регистр адреса EEAR (EEPROM Address Register) физически размещается в двух PBB — EEARH:EEARL. В этот регистр загружается адрес ячейки, к которой будет производиться обращение. Регистр адреса доступен как для записи, так и для чтения. Причем в регистре EEARH используются только младшие биты (количество задействованных битов зависит от объема EEPROM-памяти). Незадействованные биты регистра EEARH доступны только для чтения и содержат нули.

Во время записи в регистр данных EEDR (EEPROM Data Register) заносятся значения, которые будут сохранены в EEPROM по адресу, определяемому регистром EEAR, а во время чтения в этот регистр помещаются данные, считанные из EEPROM.

Регистр управления EECR (EEPROM Control Register) используется для управления доступом к EEPROM-памяти.

2. СИСТЕМА КОМАНД AVR-МИКРОКОНТРОЛЛЕРОВ

2.1. Особенности системы команд

Система команд микроконтроллеров AVR семейства Mega весьма развита и насчитывает в различных моделях от 130 до 135 различных инструкций. Несмотря на то, что микроконтроллеры AVR являются микроконтроллерами с RISC-архитектурой (процессор с сокращенным набором команд), по количеству реализованных инструкций и их разнообразию они больше похожи на микроконтроллеры с CISC-архитектурой (процессор с полным набором команд). Практически каждая из команд (за исключением команд, у которых одним из операндов является 16-битный адрес) занимает одну ячейку памяти программ. Причем это достигнуто не за счет сокращения числа команд процессора, а за счет увеличения разрядности памяти программ.

Программа для любого микроконтроллера представляет собой последовательность команд, записанных в памяти программ. Большинство команд при выполнении изменяют содержимое одного или нескольких регистров общего назначения, регистров ввода/вывода или ячеек ОЗУ.

Доступ к регистрам ввода/вывода осуществляется по их адресам, являющихся операндами команды. Однако при написании программ гораздо удобнее обращаться к регистрам, используя вместо числовых значений адресов их стандартные, принятые в документации символические имена. Чтобы задать соответствие этих имен реальным адресам, необходимо подключить в начале программы (при помощи директивы ассемблера INCLUDE) файл определения адресов регистров ввода/вывода. Помимо всего прочего, такое решение облегчит перенос программного обеспечения с одного типа кристалла на другой.

Эти файлы (для каждой модели микроконтроллеров семейства) свободно распространяются фирмой Atmel вместе с документацией на микроконтроллеры (в частности, включаемые файлы для всех выпускаемых микроконтроллеров AVR входят в комплект бесплатно распространяемой интегрированной среды AVRStudio). Для РОН, используемых в индексных регистрах, в этих файлах определяются также дополнительные символические имена XH, XL, YH, YL, ZH, ZL.

Названия этих включаемых файлов унифицированы и определяются следующим образом: <номер_модели>def.inc

Например, программа для микроконтроллера ATmega8535 должна содержать следующую директиву ассемблера:

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

Необходимо только помнить, что если для обращения к регистру ввода/вывода используются команды обмена с ОЗУ, то к символическому имени требуется прибавить число $20.

В ряде случаев значение операнда-источника может содержаться непосредственно в коде операции, а не в регистре. Это происходит в том случае, когда операндом-источником является константа.

2.2. Способы адресации

Микроконтроллеры AVR семейства Mega поддерживают 8 способов адресации для доступа к различным областям памяти данных (РОН, РВВ, ОЗУ).

4 способа представляют собой разновидности прямой адресации: прямая адресация одного РОН, прямая адресация двух РОН, прямая адресация РВВ, прямая адресация ОЗУ. В кодах команд, оперирующих с РОН и РВВ, указываются соответствующие 5-битные и 6-битные коды регистров. Команды с прямой адресацией ОЗУ – LDS и STS обеспечивают пересылку байта между одним из РОН и ячейкой ОЗУ и занимают в памяти программ два слова (32 бита), в первом из которых содержится код операции и адрес регистра общего назначения, а во втором находится адрес ячейки памяти.

При косвенной адресации адрес ячейки памяти находится в одном из индексных регистров X, Y и Z. В зависимости от дополнительных манипуляций, которые производятся над содержимым индексного регистра, различают 4 разновидности косвенной адресации: простая косвенная адресация, относительная косвенная адресация, косвенная адресация с преддекрементом и косвенная адресация с постинкрементом.

При использовании команд простой косвенной адресации обращение производится к ячейке памяти, адрес которой находится в индексном регистре. Никаких действий с содержимым индексного регистра при этом не производится. Микроконтроллеры поддерживают 6 команд (по 2 для каждого индексного регистра) простой косвенной адресации: LD Rd, X/Y/Z (пересылка байта из ОЗУ в РОН) и ST X/Y/Z, Rd (пересылка байта из РОН в ОЗУ).

При использовании команд относительной косвенной адресации адрес ячейки памяти, к которой производится обращение, получается суммированием содержимого индексного регистра (Y или Z) и константы, задаваемой в команде. Другими словами, производится обращение по адресу, указанному в команде, относительно адреса, находящегося в индексном регистре. Микроконтроллеры семейства Mega поддерживают 4 команды относительной косвенной адресации (две — для регистра Y и две — для регистра Z): LDD Rd, Y+q/Z+q (пересылка байта из ОЗУ в РОН) и ST Y+q/Z+q, Rr (пересылка байта из РОН в ОЗУ). Поскольку под значение смещения в коде команды отводится только 6 битов, оно не может превышать 64.

При использовании команд косвенной адресации с преддекрементом содержимое индексного регистра сначала уменьшается на 1, а затем производится обращение по полученному адресу. Микроконтроллеры семейства поддерживают 6 команд (по 2 для каждого индексного регистра) косвенной адресации с преддекрементом: LD Rd, -X/-Y/-Z (пересылка байта из ОЗУ в РОН) и ST -X/-Y/-Z, Rd (пересылка байта из РОН в ОЗУ).

При использовании команд косвенной адресации с постинкрементом после обращения по адресу, который находится в индексном регистре, содержимое индексного регистра увеличивается на 1. Микроконтроллеры семейства поддерживают 6 команд (по 2 для каждого индексного регистра) косвенной адресации с постинкрементом: LD Rd, X+/Y+/Z+(пересылка байта из ОЗУ в РОН) и ST X+/Y+/Z+, Rd (пересылка байта из РОН в ОЗУ).

В командах передачи управления также могут использоваться различные способы адресации памяти программ. Так, все микроконтроллеры семейства Mega имеют три команды безусловного перехода: команду относительного перехода RJMP, команду абсолютного перехода JMP, а также команду косвенного перехода IJMP.

При выполнении команды относительного перехода содержимое счетчика команд изменяется прибавлением к нему или вычитанием из него некоторого значения, являющегося операндом команды. Поскольку под значение операнда в слове команды отводится всего 12 битов, с помощью этой команды можно переходить только в пределах –2047 ... +2048 слов. В программах в качестве операндов этой команды вместо констант используются метки. Ассемблер сам вычисляет величину перехода и подставляет это значение в слово команды.

При выполнении команды абсолютного перехода в счетчик команд просто загружается значение нового адреса, являющееся операндом команды. Эта команда имеет длину в два машинных слова. Поскольку максимальное значение адреса равно $3FFFFF, то с помощью этой команды можно осуществлять переход в пределах всего адресного пространства имеющихся на сегодняшний день микроконтроллеров AVR.

При выполнении команды косвенного перехода осуществляется переход по адресу, который находится в индексном регистре Z. Соответственно, процесс выполнения команды сводится к загрузке содержимого индексного регистра в счетчик команд. Так как индексный регистр — 16-битный, то максимально возможная величина перехода составляет 128 Кбайт (в моделях с объемом памяти программ более 128 Кбайт для выполнения перехода в пределах всего адресного пространства используется дополнительный регистр для задания старшей части адреса).

Как и для реализации безусловных переходов, для вызова подпрограмм в микроконтроллерах семейства Mega имеются три команды: команда относительного вызова RCALL, команда абсолютного вызова CALL и команда косвенного вызова ICALL. Эти команды работают так же, как и соответствующие команды безусловного перехода, но перед выполнением перехода запоминается в стеке адрес возврата из подпрограммы (текущее значение счетчика команд). В конце каждой подпрограммы должна находиться команда возврата из нее. В системе команд микроконтроллеров семейства имеется две таких команды. Для возврата из обычной подпрограммы, вызываемой командами вызова подпрограмм, используется команда RET. Для возврата из подпрограммы обработки прерывания используется команда RETI. Обе команды восстанавливают из стека содержимое счетчика команд, сохраненное там перед переходом к подпрограмме. Команда возврата из подпрограммы RETI дополнительно устанавливает в 1 флаг общего разрешения прерываний регистра состояния SREG, сбрасываемый аппаратно при возникновении прерывания.

В командах условного перехода AVR-микроконтроллеров используется только относительная адресация, а под значение смещения в слове команды отводится всего 7 битов, поэтому максимальная величина перехода составляет от –63 до +64 слов. Кроме того, имеются команды типа «проверка/пропуск», в ко-

торых производится проверка условия, результат которой влияет на выполнение следующей команды. Если условие истинно, следующая команда игнорируется.

2.3. Типы и виды команд

Все множество команд микроконтроллеров AVR семейства Mega можно разбить на несколько групп:

- команды логических операций;
- команды арифметических операций и команды сдвига;
- команды операций с битами;
- команды пересылки данных;
- команды передачи управления;
- команды управления системой.

Далее коротко описана каждая группа команд.

Команды логических операций (табл. 2) позволяют выполнять стандартные логические операции над байтами, такие как логическое умножение (И), логическое сложение (ИЛИ), операцию «Исключающее ИЛИ», а также вычисление обратного (дополнение до единицы) и дополнительного (дополнение до двух) кодов числа. К этой группе можно отнести также команды очистки/установки регистров и команду перестановки полубайтов. Операции производятся между регистрами общего назначения, либо между регистром и константой; результат сохраняется в РОН. Все команды из этой группы выполняются за один такт.

Таблица 2. Логические команды

Мнемоника	Описание	Операция	Флаги
AND Rd, Rr	«Логическое И» двух РОН	Rd = Rd • Rr	Z,N,V
ANDI Rd, K *	«Логическое И» РОН и константы	Rd = Rd • K	Z,N,V
EOR Rd, Rr	«Исключающее ИЛИ» двух РОН	Rd = Rd⊕Rr	Z,N,V
OR Rd, Rr	«Логическое ИЛИ» двух РОН	Rd = Rd v Rr	Z,N,V
ORI Rd, K *	«Логическое ИЛИ» РОН и константы	Rd = Rd v K	Z,N,V
COM Rd	Перевод в обратный код	Rd = $FF - Rd	Z,C,N,V
NEG Rd	Перевод в дополнительный код	Rd = $00 - Rd	Z,C,N,V,H
CLR Rd	Сброс всех битов РОН	Rd = Rd⊕Rd	Z,N,V
SER Rd *	Установка всех битов РОН	Rd = $FF	-
TST Rd	Проверка РОН на отрицательное или нулевое значение	Rd • Rd	Z,N,V
SWAP Rd	Обмен местами полубайтов в РОН	Rd(3...0) = Rd(7...4), Rd(7...4) = Rd(3...0)	-
$0 \le d \le 31$, $0 \le r \le 31$, $0 \le K \le 255$ (* в командах ANDI, ORI и SER: $16 \le d \le 31$)			

К группе команд арифметических операций и сдвига (табл. 3) относятся команды, позволяющие выполнять такие базовые операции, как сложение, вычитание, сдвиг (вправо и влево), инкрементирование, декрементирование, а также умножение. Все операции производятся только над регистрами общего назначения. При этом микроконтроллеры AVR позволяют легко оперировать

как знаковыми, так и беззнаковыми числами, а также работать с числами, представленными в дополнительном коде. Почти все команды рассматриваемой группы выполняются за один такт. Команды умножения и команды, оперирующие 2-байтными значениями, выполняются за два такта.

Таблица 3. Арифметические и сдвиговые команды

Мнемоника	Описание	Операция	Флаги
ADD Rd, Rr	Сложение двух РОН	Rd = Rd + Rr	Z,C,N,V,H
ADC Rd, Rr	Сложение двух РОН с переносом	Rd = Rd + Rr + C	Z,C,N,V,H
ADIW Rd, K **	Сложение регистровой пары с константой	Rd+1:Rd = Rd+1:Rd + K	Z,C,N,V,S
SUB Rd, Rr	Вычитание двух РОН	Rd = Rd - Rr	Z,C,N,V,H
SUBI Rd, K *	Вычитание константы из РОН	Rd = Rd - K	Z,C,N,V,H
SBC Rd, Rr	Вычитание двух РОН с заемом	Rd = Rd – Rr - C	Z,C,N,V,H
SBCI Rd, K *	Вычитание константы из РОН с заемом	Rd = Rd – K - C	Z,C,N,V,H
SBIW Rd, K **	Вычитание константы из регистровой пары	Rd+1:Rd = Rd+1:Rd - K	Z,C,N,V,S
DEC Rd	Декрементирование РОН	Rd = Rd - 1	Z,N,V
INC Rd	Инкрементирование РОН	Rd = Rd + 1	Z,N,V
ASR Rd	Арифметический сдвиг вправо	Rd(n) = Rd(n+1), n = 0..6	Z,C,N,V
LSL Rd	Логический сдвиг влево	Rd(n+1) = Rd(n), Rd(0) = 0	Z,C,N,V
LSR Rd	Логический сдвиг вправо	Rd(n) = Rd(n+l), Rd(7) = 0	Z,C,N,V
ROL Rd	Сдвиг влево через перенос	Rd(0) = C, Rd(n+l) = Rd(n), C = Rd(7)	Z,C,N,V
ROR Rd	Сдвиг вправо через перенос	Rd(7) = C, Rd(n) = Rd(n+l), C = Rd(0)	Z,C,N,V
MUL Rd, Rr	Умножение беззнаковых чисел	R1:R0 = Rd*Rr	Z,C
MULS Rd, Rr *	Умножение чисел со знаком	R1:R0 = Rd*Rr	Z,C
MULSU Rd, Rr ***	Умножение беззнакового числа на число со знаком	R1:R0 = Rd*Rr	Z,C
FMUL Rd, Rr ***	Умножение дробных беззнаковых чисел	R1:R0 = (Rd*Rr) << l	Z,C
FMULS Rd, Rr ***	Умножение дробных чисел со знаком	R1:R0 = (Rd*Rr) << l	Z,C
FMULSU Rd, Rr ***	Умножение дробного беззнакового числа и дробного числа со знаком	R1:R0 = (Rd*Rr) << l	Z,C
$0 \le d \le 31$, $0 \le r \le 31$, $0 \le K \le 255$ (* в командах SUBI, SBCI, MULS: $16 \le d \le 31$, $16 \le r \le 31$; ** в командах ADIW, SBIW: d = {24, 26, 28, 30}, $0 \le K \le 63$; *** в командах MULSU, FMUL, FMULS, FMULSU $16 \le d \le 23$, $16 \le r \le 23$)			

К группе команды битовых операций (табл. 4) относятся команды, выполняющие установку или сброс заданного бита РОН или РВВ. Причем для из-

менения битов регистра состояния SREG имеются отдельные команды (точнее говоря, эквивалентные мнемонические обозначения общих команд), так как проверка состояния битов именно этого регистра производится чаще всего.

Таблица 4. Команды битовых операций

Мнемоника	Описание	Операция	Флаги
CBR Rd, K *	Сброс бита(ов) РОН	Rd = Rd • ($FF - K)	Z, N, V
SBR Rd, K *	Установка бита(ов) РОН	Rd = Rd v K	Z, N, V
CBI P, b	Сброс бита РВВ	P.b = 0	-
SBI P, b	Установка бита РВВ	P.b = 1	-
BCLR s	Сброс флага	SREG.s = 0	SREG.s
BSET s	Установка флага	SREG.s = 1	SREG.s
BLD Rd, b	Загрузка бита РОН из флага T (SREG)	Rd.b = T	-
BST Rr, b	Запись бита РОН в флаг T (SREG)	T = Rr.b	T
CLC	Сброс флага переноса	C = 0	C
SEC	Установка флага переноса	C = 1	C
CLN	Сброс флага отрицательного числа	N = 0	N
SEN	Установка флага отрицательного числа	N = 1	N
CLZ	Сброс флага нуля	Z = 0	Z
SEZ	Установка флага нуля	Z = 1	Z
CLI	Общее запрещение прерываний	I = 0	I
SEI	Общее разрешение прерываний	I = 1	I
CLS	Сброс флага знака	S = 0	S
SES	Установка флага знака	S = 1	S
CLV	Сброс флага переполнения дополнительного кода	V = 0	V
SEV	Установка флага переполнения дополнительного кода	V = l	V
CLT	Сброс флага T	T = 0	T
SET	Установка флага T	T = l	T
CLH	Сброс флага половинного переноса	H = 0	H
SEH	Установка флага половинного переноса	H = l	H
0 ≤ d ≤ 31, 0 ≤ r ≤ 31, 0 ≤ b ≤ 7, 0 ≤ s ≤ 7, 0 ≤ K ≤ 255 (* в командах CBR, SBR: 16 ≤ d ≤ 31)			

Все задействованные биты РВВ имеют свои символические имена. Определения этих имен описаны в том же включаемом файле, что и определения символических имен адресов регистров. Соответственно после включения в программу указанного файла в командах вместо числовых значений номеров битов можно будет указывать их символические имена.

Следует помнить, что в командах CBR и SBR операндом является битовая маска, а не номер бита. Для получения битовой маски из номера бита следует воспользоваться ассемблерным оператором «сдвиг влево» («<<»), как показано в следующем примере:

```
sbr rl6, (1<<SE) + (1<<SM)
out MCUCR, rl6      ; Установить флаги SE и SM регистра MCUCR
```

Всем командам данной группы требуется один такт для выполнения, за исключением команд установки/сброса бита РВВ, выполняемых за 2 такта.

Команды пересылки данных (табл. 5) предназначены для пересылки содержимого ячеек, находящихся в адресном пространстве памяти данных.

Таблица 5. Команды пересылки данных

Мнемоника	Описание	Операция	Флаги
MOV Rd, Rr	Пересылка между РОН	Rd = Rr	-
MOVW Rd, Rr	Пересылка 2-байтных значений	Rd+l:Rd = Rr+l:Rr	-
LDI Rd, K *	Загрузка константы в РОН	Rd = K	-
LD Rd, X	Косвенное чтение	Rd = [X]	-
LD Rd, X+	Косвенное чтение с постинкрементом	Rd = [X], X = X + l	-
LD Rd, -X	Косвенное чтение с преддекрементом	X = X - l, Rd = [X]	-
LD Rd, Y	Косвенное чтение	Rd = [Y]	-
LD Rd, Y+	Косвенное чтение с постинкрементом	Rd = [Y], Y = Y + l	-
LD Rd, -Y	Косвенное чтение с преддекрементом	Y = Y - l, Rd = [Y]	-
LDD Rd, Y+q	Косвенное относительное чтение	Rd= [Y + q]	-
LD Rd, Z	Косвенное чтение	Rd = [Z]	-
LD Rd, Z+	Косвенное чтение с постинкрементом	Rd = [Z], Z = Z +l	-
LD Rd, -Z	Косвенное чтение с преддекрементом	Z = Z - l, Rd = [Z]	-
LDD Rd, Z+q	Косвенное относительное чтение	Rd = [Z + q]	-
LDS Rd, k	Непосредственное чтение из ОЗУ	Rd = [k]	-
ST X, Rr	Косвенная запись	[X] = Rr	-
ST X+, Rr	Косвенная запись с постинкрементом	[X] = Rr, X = X + l	-
ST -X, Rr	Косвенная запись с преддекрементом	X = X - l, [X] = Rr	-
ST Y, Rr	Косвенная запись	[Y] = Rr	-
ST Y+, Rr	Косвенная запись с постинкрементом	[Y] = Rr, Y = Y + l	-
ST -Y, Rr	Косвенная запись с преддекрементом	Y = Y - l, [X] = Rr	-
STD Y+q, Rr	Косвенная относительная запись	[Y + q] = Rr	-
ST Z, Rr	Косвенная запись	[Z] = Rr	-
ST Z+, Rr	Косвенная запись с постинкрементом	[Z] = Rr, Z = Z + l	-
ST -Z, Rr	Косвенная запись с преддекрементом	Z = Z - l, [Z] = Rr	-
STD Z+q, Rr	Косвенная относительная запись	[Z + q] = Rr	-
STS k, Rr	Непосредственная запись в ОЗУ	[k] = Rr	-
LPM	Загрузка данных из памяти программ	R0 = {Z}	-
LPM Rd, Z	Загрузка данных из памяти программ	Rd = {Z}	-
LPM Rd, Z+	Загрузка данных из памяти программ	Rd = {Z}, Z = Z + 1	-
SPM	Запись в память программ	{Z} = R1:R0	-
IN Rd, P	Пересылка из РВВ в РОН	Rd = P	-
OUT P, Rr	Пересылка из РОН в РВВ	P = Rr	-
PUSH Rr	Сохранение байта в стеке	STACK = Rr	-
POP Rd	Извлечение байта из стека	Rd = STACK	-
$0 \le d \le 31$, $0 \le r \le 31$, $0 \le K \le 255$, $0 \le k \le 65535$, $0 \le q \le 63$, $0 \le P \le 63$ (* в команде LDI: $16 \le d \le 31$); [] – содержимое памяти данных; {} – содержимое памяти команд			

Разделение адресного пространства на три части (РОН, РВВ, ОЗУ) предопределило разнообразие команд данной группы. Пересылка данных, выполняемая командами группы, может производиться в следующих направлениях:

• РОН <=> РОН;

• РОН <=> РВВ;

• РОН <=> память данных.

Также к данной группе можно отнести стековые команды PUSH и POP, позволяющие сохранять в стеке и восстанавливать из стека содержимое РОН.

На выполнение команд данной группы требуется от одного до трех тактов в зависимости от команды.

В группу команд передачи управления (табл. 6) входят команды перехода, вызова подпрограмм и возврата из них и команды типа «проверка/пропуск», пропускающие следующую за ними команду при выполнении некоторого условия. Также к этой группе относят команды сравнения, формирующие флаги регистра SREG и предназначенные, как правило, для работы совместно с командами условного перехода. Все команды условного перехода можно разбить на две подгруппы. Первая подгруппа — команды условного перехода общего назначения. В эту подгруппу входят две команды — BRBS s, k и BRBC s, k, в которых явно задается номер тестируемого флага регистра SREG. Соответственно, переход осуществляется при SREG.s = 0 (BRBC) или SREG.s = 1 (BRBS).

Таблица 6. Команды передачи управления

Мнемоника	Описание	Операция	Флаги
RJMP k **	Относительный безусловный переход	PC = PC + к + 1	-
IJMP	Косвенный безусловный переход	PC = Z	-
JMP k ***	Абсолютный переход	PC = k	-
RCALL k **	Относительный вызов подпрограммы	STACK = PC, PC = PC + k + 1	-
ICALL	Косвенный вызов подпрограммы	STACK = PC, PC = Z	-
CALL k ***	Абсолютный вызов подпрограммы	STACK = PC, PC = k	-
RET	Возврат из подпрограммы	PC = STACK	-
RETI	Возврат из п/п обработки прерывания	PC = STACK	I
CP Rd, Rr	Сравнение РОН	Rd - Rr	Z,N,V,C,H
CPC Rd, Rr	Сравнение РОН с учетом переноса	Rd – Rr - C	Z,N,V,C,H
CPI Rd, K *	Сравнение РОН с константой	Rd - K	Z,N,V,C,H
CPSE Rd, Rr	Сравнение и пропуск следующей команды при равенстве	Если Rd = Rr, то PC = PC + 2(3)	-
SBRC Rr, b	Пропуск следующей команды, если бит РОН сброшен	Если Rr.b = 0, то PC = PC + 2(3)	-
SBRS Rr, b	Пропуск следующей команды, если бит РОН установлен	Если Rr.b= 1, то PC = PC + 2(3)	-
SBIC P, b	Пропуск следующей команды, если бит РВВ сброшен	Если P.b = 0, то PC = PC + 2(3)	-
SBIS P, b	Пропуск следующей команды, если бит РВВ установлен	Если P.b = 1, то PC = PC + 2(3)	-
BRBC s, k	Переход, если флаг s регистра SREG	Если SREG.s = 0, то	-

	сброшен	PC = PC + k + 1	

Окончание табл. 6.

Мнемоника	Описание	Операция	Флаги
BRBS s, k	Переход, если флаг s регистра SREG установлен	Если SREG.s = l, то PC = PC + k + 1	-
BRCS k	Переход по переносу	Если C = 1, то PC = PC + k + 1	-
BRCC k	Переход, если нет переноса	Если C = 0, то PC = PC + k + 1	-
BREQ k	Переход по «равно»	Если Z = 1, то PC = PC + k + 1	-
BRNE k	Переход по «не равно»	Если Z = 0, то PC = PC + k + 1	-
BRSH k	Переход по «больше или равно»	Если C = 0, то PC = PC + k + 1	-
BRLO k	Переход по «меньше»	Если C =1, то PC = PC + k + 1	-
BRMI k	Переход по «отрицательное значение»	Если N = 1, то PC = PC + k + 1	-
BRPL k	Переход по «положительное значение»	Если N = 0, то PC = PC + k + 1	-
BRGE k	Переход по «больше или равно» (числа со знаком)	Если (N⊕V) = 0, то PC = PC + k + 1	-
BRLT k	Переход по «меньше нуля» (числа со знаком)	Если (N⊕V) = l, то PC = PC + k + 1	-
BRHS k	Переход по половинному переносу	Если H = 1, то PC = PC + k + 1	-
BRHC k	Переход, если нет половинного переноса	Если H = 0, то PC = PC + k + 1	-
BRTS k	Переход, если флаг T установлен	Если T = 1, то PC = PC + k + 1	-
BRTC k	Переход, если флаг T сброшен	Если T = 0, то PC = PC + k + 1	-
BRVS k	Переход по переполнению дополнительного кода	Если V = 1, то PC = PC + k + 1	-
BRVC k	Переход, если нет переполнения дополнительного кода	Если V = 0, то PC = PC + k + 1	-
BRID k	Переход, если прерывания запрещены	Если I = 0, то PC = PC + k + 1	-
BRIE k	Переход, если прерывания разрешены	Если I = 1, то PC = PC + k + 1	-
$0 \le d \le 31$, $0 \le r \le 31$, $0 \le b \le 7$, $0 \le s \le 7$, $0 \le K \le 255$, $-64 \le k \le 63$, $0 \le P \le 31$ (* в команде CPI: $16 \le d \le 31$; ** в командах RJMP, RCALL: $-2048 \le k \le +2047$; *** в командах JMP, CALL: $0 \le k \le 4M$)			

Другую подгруппу составляют 18 специализированных команд, каждая из которых выполняет переход по какому-либо конкретному условию («равно», «больше или равно», «был перенос» и т. п.). Причем одни команды используются после сравнения беззнаковых чисел, другие — после сравнения чисел со знаком. Вообще говоря, эти команды являются лишь эквивалентными мнемо-

ническими обозначениями команд BRBS s, k и BRBC s, k с определенными значениями операнда s. Команда BREQ k («переход, если равно») имеет, например, такой же код операции, что и команда BRBS 1, k, а команда BRGE k («переход, если больше или равно для чисел со знаком») — BRBC 4 , k.

Команды передачи управления нарушают линейное выполнение основной программы, поэтому при каждом выполнении команды этой группы (кроме команд сравнения) нормальное функционирование конвейера нарушается. В результате на выполнение команд затрачивается больше одного такта.

В группу команд управления системой входят всего 4 команды:

• NOP — пустая команда;

• SLEEP — перевод микроконтроллера в режим пониженного энергопотребления;

• WDR — сброс сторожевого таймера;

• BREAK — команда, используемая внутрисхемным отладчиком.

Все команды этой группы, кроме последней, выполняются за один такт.

3. ЛАБОРАТОРНЫЙ КОМПЛЕКС «МИКРОКОНТРОЛЛЕРЫ И АВТОМАТИЗАЦИЯ»

3.1. Структура и принципы работы комплекса

Лабораторный комплекс «Микроконтроллеры и автоматизация» предназначен для обучения студентов различных специальностей, изучающих дисциплины по автоматизации различных отраслей производства, программированию интегральных микроконтроллеров (однокристальных микро-ЭВМ). Комплекс включает центральный персональный компьютер (ноутбук) со специальным программным обеспечением и 8 учебных микроконтроллерных модулей на базе AVR-микроконтроллеров ATmega8535 (рис. 4). К каждому модулю подключается клавиатура для ввода и редактирования текста программ. Питание рабочих мест осуществляется от блоков питания БП через блоки связи с компьютером (USB-хабы) по кабелям рабочих мест.

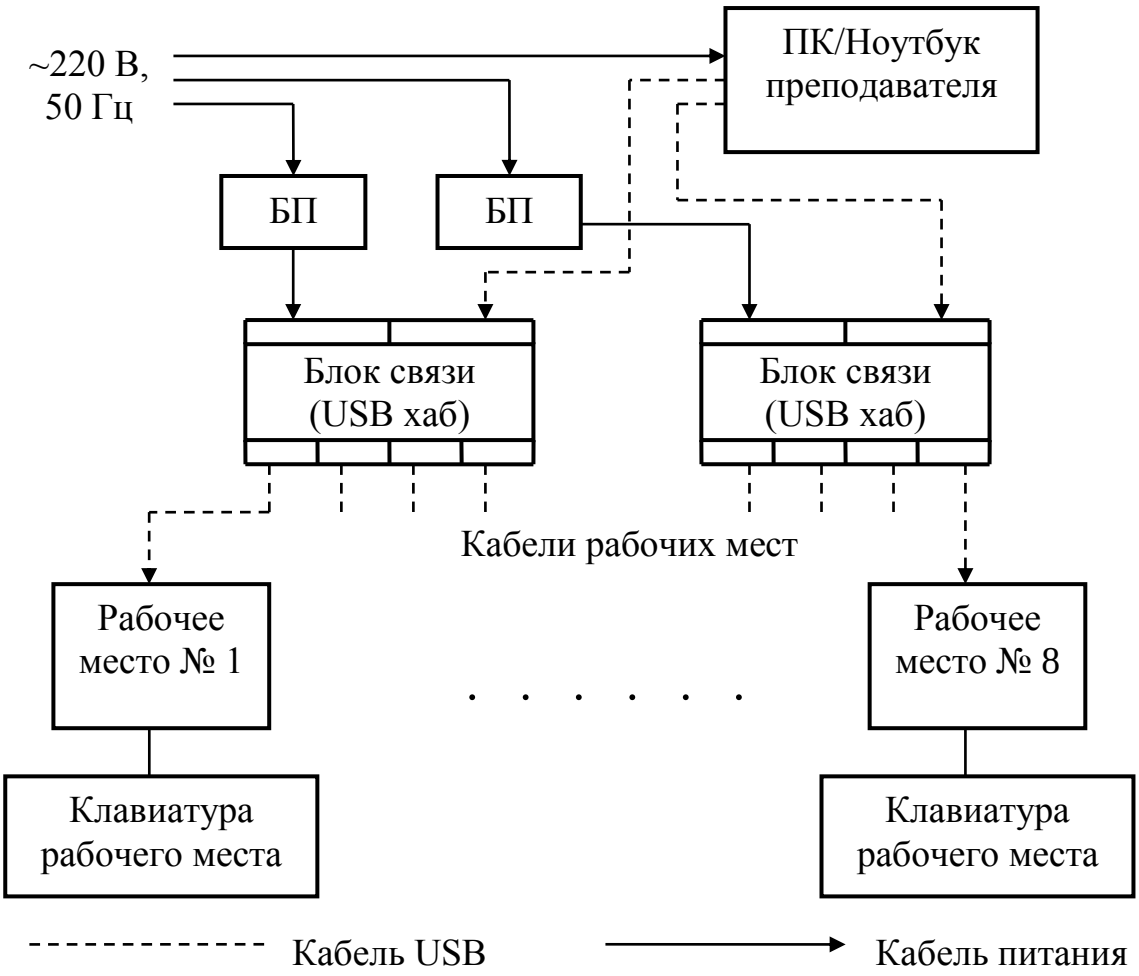


Рис. 4. Функциональная схема лабораторного комплекса

Студент на рабочем месте набирает программу как при обычной работе на персональном компьютере, но контролирует набор программы не по монитору, а по жидкокристаллическому индикатору (ЖКИ), установленному в мо-

дуле рабочего места. В процессе набора программы необходимо перевести тумблер «Режим» учебного модуля в положение «Ред».

На мониторе преподавательского компьютера в восьми рабочих окнах (по чиклу рабочих мест) индицируются тексты программ, набираемые студентами на рабочих местах. Преподаватель имеет возможность отслеживать работу студентов, проверять работоспособность набранных ими программ, а также помогать студентам в работе. Создаваемые программы сохраняются в памяти компьютера преподавателя.

Готовая программа может быть отправлена на компиляцию как со студенческого рабочего места, так и с преподавательского. Если ошибок в тексте программы нет и компиляция завершилась успешно, то запускается процесс записи программы во флэш-память микроконтроллера. Процесс записи программы индицируется на экране рабочего места, при этом горит светодиод «Блокировка» сигнализируя о том, что редактирование программы на время процесса компиляции и программирования отключено. Если светодиод «Блокировка» в процессе компиляции программы студентом несколько раз мигает, это сигнал наличия ошибок в тексте программы.

Светодиод «Блокировка» включается также при редактировании текста программы преподавателем или в режиме демонстрации преподавателем примера программ всем рабочим местам.

Функционирование запрограммированного микроконтроллера может быть проверено при переключении тумблера «Режим» учебного модуля в положение «Работа».

Каждый вариант лабораторной работы предусматривает ввод каких-либо команд в микроконтроллерный модуль в виде нажатия – отпускания кнопок или включения – выключения тумблеров и наблюдение реакции на эти команды по загоранию – потуханию светодиодов, включению – изменению тона звукогенератора или индикации информации на семисегментных индикаторах. Например, поставлена задача: «Реализовать сложение двух чисел. При нажатии одной кнопки на индикаторах высвечивается первое слагаемое, при нажатии второй кнопки – второе слагаемое, при нажатии третьей кнопки дается команда на выполнение операции сложения. Нажатие четвертой кнопки выводит на индикацию содержимое ячейки результата как до, так и после операции сложения». В данной задаче используются четыре кнопки и индикаторы. Необходимо выбрать, какие кнопки будут использованы, т.е. определить их адреса, и разобраться с адресацией индикаторов и принципом управления ими.

В схеме учебного модуля (рис. 5) в качестве органов управления использованы восемь тумблеров, которые в верхнем положении фиксируются, а в нижнем положении не имеют фиксации. В среднем положении тумблера на соответствующие выводы микроконтроллера подаётся уровень логического нуля, а в верхнем и нижнем положениях – уровень логической единицы.

Вращением движка потенциометра обеспечивается изменение значения напряжения Uвх на входе аналого-цифрового преобразователя.

Рис. 5. Внешний вид модуля микроконтроллера

В качестве выходных элементов используются восемь светодиодов VD1...VD8, звукоизлучатель HA1 и четыре семисегментных индикатора HG1...HG4.

На лицевой панели блока управления рядом с каждым элементом указана его адресация. Например, светодиод VD5 имеет адрес PORTB.4 (PB4), а сегмент «d» индикаторов – PORTC.3 (PC3).

На рисунке 6 представлены обозначения сегментов индикаторов a...h, которые имеют соответственно адреса PORTC.0...PORTC.7.

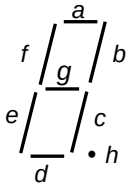


Рис. 6. Семисегментный индикатор

Лабораторный комплекс может также использоваться для микроконтроллерного управления виртуальными объектами, которые имитируются программно с помощью персонального компьютера. В этом случае тумблер «Режим» учебного модуля необходимо переключить в положение «Авт». В лабораторном комплексе реализованы четыре варианта виртуальных объектов.

3.2. Ввод и редактирование программы

Включение лабораторного комплекса осуществляется включением питания учебных модулей и преподавательского ноутбука, после чего запускается программное обеспечение комплекса (через ярлык «8 местный стенд Микроконтроллер» на рабочем столе). При этом возможны два варианта: начать новую работу или открыть один из существующих проектов.

В случае начала новой работы имеется возможность добавить комментарии к каждому рабочему месту (например, фамилии учащегося), изменить дату проведения работы (по умолчанию текущая). При открытии одного из существующих проектов можно выбрать любой прошлый проект, перемещаясь по папкам проектов, датированных в хронологическом порядке. В этом случае на экране ноутбука появятся 8 окон с набранным в прошлый раз текстом программ. Этот же текст отобразится и на индикаторах микроконтроллерных модулей. Можно продолжать редактирование (тумблер «Режим» в положении «Ред»).

Пример содержимого экрана компьютера во время работы программы представлен на рисунке 7.

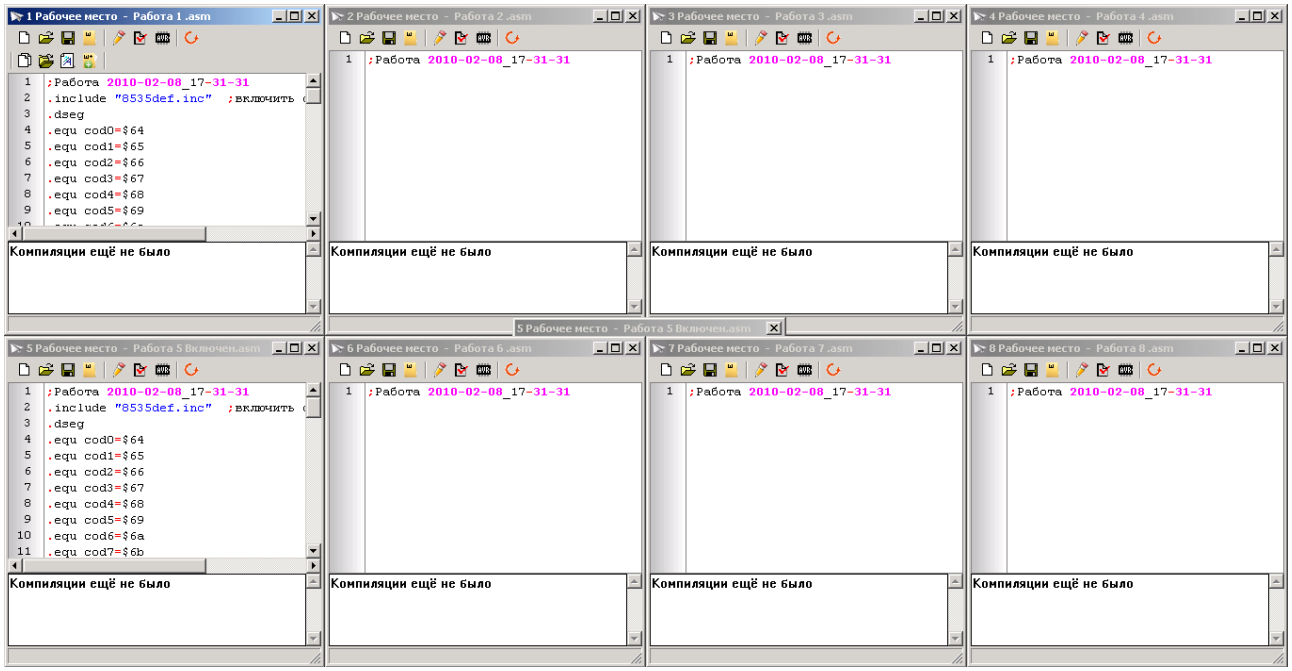


Рис. 7. Пример содержимого экрана компьютера во время работы программы

Назначение "горячих'" клавиш, доступных студенту на клавиатуре учебного микропроцессорного модуля в процессе ввода и редактирования программ, представлено в таблице 7.

Таблица 7. «Горячие» клавиши

Ctrl + Z	Откатка текста программы на несколько шагов назад
Ctrl + X	Вырезание выделенного построчно текста в буфер обмена
Ctrl + C	Копирование выделенного построчно текста в буфер обмена
Ctrl + V	Вставка текста из буфера обмена в позицию курсора
Ctrl + Вниз	Выделение строки под курсором и перемещение вниз. Выделенные строки медленно мигают. Допускается выделение несплошного блока
Ctrl + Вверх	Выделение строки под курсором и перемещение вверх. Выделенные строки медленно мигают. Допускается выделение несплошного блока
F2	Запись программы на винчестер
F9	Компиляция и прошивка программы. В процессе компиляции текста программы студентом, если светодиод «Блокировка» несколько раз мигает – это сигнал наличия ошибок в тексте программы. После успешной компиляции сразу запускается процесс прошивки, он индицируется на ЖКИ индикаторе
P5/F6	Показать ошибки /вернуться к редактированию
F7	Перейти к строке с ошибкой
F11/F12	Показать номера строк /убрать номера строк
Ctrl + Shift или Alt + Shift	Переход на русский язык и обратно на английский
PageUp	3 строки вверх
PageDown	3 строки вниз
Home	Начало строки
End	Конец строки
Delete	Удалить символ под курсором
Backspace	Удалить символ со сдвигом влево
Стрелки	Управление курсором

Преподаватель может контролировать текст программы на наличие ошибок, нажав кнопку проверки. Он также может перевести комплекс в демонстрационный режим, загрузив на первое рабочее место пример программы и нажав кнопку демонстрационного режима. Текст программы и курсор-указатель будут транслироваться на все рабочие места, на которых загорается светодиод «Блокировка». Можно загрузить подготовленный текст (пример) какой-либо программы на отдельные или все рабочие места. В случае необходимости можно очистить текст на рабочих местах кнопкой очистки.

Если предстоит лабораторная работа по программированию студентами виртуальных объектов, то преподавателю необходимо на рабочие места студен-

тов загрузить шаблоны программ для работы с виртуальными объектами. Для этого преподаватель нажимает кнопку загрузки шаблона на отдельное рабочее место или на все рабочие места и выбирает нужный шаблон для виртуальных объектов .asm.

Студент может самостоятельно компилировать программу по клавише F9. При наличии ошибок в тексте программы у преподавателя в окне соответствующего рабочего места появляется список ошибок. На рабочем месте студента список допущенных ошибок можно увидеть по нажатию клавиши F5. При нажатии клавиши F6 на ЖКИ снова индицируется программа. Для перехода к строке с ошибкой необходимо нажать F7. После успешной компиляции сразу запускается процесс прошивки программы.

3.3. Проверка функционирования программы

Тумблер «Режим» нужно переключить в положение «Работа». Кнопками и тумблерами, в зависимости от запрограммированной задачи, даются необходимые команды, и визуально контролируется работа программы.

Если в работе обнаружены ошибки, то студент вносит необходимые коррективы в свою программу. Вновь повторяются процедуры компиляции, исправления возможных ошибок, записи программы в микроконтроллер и проверка функционирования программы. Лабораторная работа выполнена, если программа выполняет все заданные функции.

Для проверки работоспособности программы управления виртуальным объектом после компиляции и записи программы в контроллер студент должен перевести тумблер «Режим» в положение «Авт». После этого у преподавателя появляется окно выбора виртуального объекта. Преподаватель должен выбрать соответствующий варианту студента объект, а затем переключить управление в режим «От контроллера». Кнопками и тумблерами, в зависимости от задачи, даются необходимые команды, и визуально контролируется работа механизмов. Если в работе обнаружены отклонения от данной последовательности, студент корректирует свою программу и аналогичным образом проверяет ее работоспособность. Для возврата к редактированию преподаватель закрывает окно с виртуальным объектом, после чего студенту необходимо перевести тумблер «Режим» в положение «Ред». При работе с комплексом нельзя одновременно запускать больше одного виртуального объекта.

4. АССЕМБЛЕР AVR-МИКРОКОНТРОЛЛЕРОВ

4.1. Особенности ассемблера

Здесь представлена основная информация по ассемблеру всей серии AVR, т.к. все микроконтроллеры этой серии программно совместимы.

Ассемблер – это инструмент, с помощью которого создаётся программа для микроконтроллера. Ассемблер транслирует ассемблируемый исходный код программы в объектный код, который может быть непосредственно введен в программную память микроконтроллера, а также использоваться в симуляторах или эмуляторах AVR.

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

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

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

Строка программы может быть в одной из четырёх форм:

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

Комментарий имеет следующую форму:

```
; [Текст]
```

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

Операнды можно задавать в различных форматах:

– десятичный (по умолчанию): 10,255
– шестнадцатеричный (два способа): 0x0a, $0a
– двоичный: 0b00001010, 0b11111111
– восьмеричный (впереди ноль): 010, 077

4.2. Директивы ассемблера

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

Директивы соответствуют второй версии компилятора ассемблера avrasm2.exe компании «Atmel».

Директивы ассемблера приведены в таблице 8.

Синтаксис всех директив следующий:

```
.[директива]
```

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

Таблица 8. Директивы ассемблера

Директива	Описание
BYTE	Зарезервировать байт под переменную
CSEG	Сегмент кодов
DB	Задать постоянным(и) байт(ы) в памяти
DEF	Задать символическое имя регистру
DEVICE	Задать для какого типа микроконтроллера компилировать
DSEG	Сегмент данных
DW	Задать постоянное(ые) слово(а) в памяти
EQU	Установить символ равный выражению
ESEG	Сегмент EEPROM
EXIT	Выход из файла
INCLUDE	Включить исходный код из другого файла
LIST	Включить генерацию .lst - файла
NO.LIST	Выключить генерацию .lst - файла
ORG	Начальный адрес программы
SET	Установить символ, равный выражению

Дадим несколько пояснений наиболее важным директивам ассемблера.

Директива CSEG указывает на начало сегмента кодов. Ассемблируемый файл может иметь несколько кодовых сегментов, которые будут объединены в один при ассемблировании. Синтаксис:

```
.CSEG
```

Пример:

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

Директива DSEG указывает на начало сегмента данных. Ассемблируемый файл может содержать несколько сегментов данных, которые потом будут собраны в один при ассемблировании. Обычно сегмент данных состоит лишь из директив BYTE и меток. Синтаксис:

```
.DSEG
```

Пример:

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

Директива ESEG указывает на начало сегмента EEPROM памяти.

Ассемблируемый файл может содержать несколько EEPROM сегментов, которые будут собраны в один сегмент при ассемблировании. 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. В данной версии комплекса программирование EEPROM может осуществляться только непосредственно из программы учащегося. Прямое программирование в процессе прошивки не предусмотрено.

Синтаксис:

```
.ESEG
```

Пример:

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

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

Синтаксис:

```
.ORG адрес
```

Пример:

```
.DSEG              ; Начало сегмента данных
.ORG 0x37          ; Установить адрес ОЗУ на 37h
variable: .BYTE 1 ; Зарезервировать байт СОЗУ по адресу 37h
.CSEG
.ORG 0x10          ; Установить счетчик команд на адрес 10h
mov r0,rl          ; Чего-нибудь делать
```

Директива DB резервирует ресурсы памяти (байты) в программной памяти или в EEPROM. Директиве должна предшествовать метка. 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.

Синтаксис:

```
LABEL: .DB список выражений
```

Пример:

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

Директива DW резервирует ресурсы памяти (слова) в программной памяти или в EEPROM. Директиве должна предшествовать метка. 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.

Синтаксис:

```
LABEL: .DW список выражений
```

Пример:

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

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

Синтаксис:

```
.DEF Имя = Регистр
```

Пример:

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

Директива EQU присваивает значение метке. Эта метка может быть использована в других выражениях. Значение этой метки нельзя изменить или переопределить.

Синтаксис:

```
.EQU метка = выражение
```

Пример:

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

Директива INCLUDE предлагает 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.

Синтаксис:

```
.INCLUDE "имя файла"
```

Пример:

```
; iodefs.asm:
.EQU sreg = 0x3f   ; Регистр статуса
.EQU sphigh = 0x3e ; Старший байт указателя стека.
.EQU splow = 0x3d; ; Младший байт указателя стека.
```

```
; incdemo.asm
.INCLUDE iodefs.asm ; Включить файл «iodefs.asm»
in r0,sreg          ; Прочитать регистр статуса
```

EXIT – выйти из файла.

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.

Синтаксис:

```
.EXIT
```

Пример:

```
.EXIT ; выйти из этого файла
```

DEVICE – указать, для какого микроконтроллера ассемблировать.

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

Синтаксис:

```
.DEVICE AT90S1200 |AT90S2313 | AT90S2323 | AT90S2333 |
AT90S2343 | AT90S4414 | AT90S4433 | AT90S4434 | AT90S8515 |
AT90S8534 | AT90S8535 | ATtinyl1 | ATtinyl2 | ATtiny22 |
ATmega64 | ATmega128| Atmega8535
```

Пример:

```
.DEVICE ATmega8535 ; использовать ATmega8535
.CSEG
.ORG 0000
jmp label1; При ассемблировании появится сообщение, что
          ; ATmega8535 не поддерживает команду jmp в
          ; таблице векторов прерываний
```

4.3. Структура ассемблерной программы

Программа, написанная на ассемблере, должна иметь определенную структуру.

Предлагается следующий шаблон (для ATmega8535)

```
;*****************************************************
; название программы,
; краткое описание, необходимые пояснения
;*****************************************************
; *********** подключаемые дополнительные файлы
.include "m8535def.inc"  ; файл описания ATmega8535
.include «имя_файла1.расширение»   ; включение дополнительных
.include «имя_файла2.расширение»  ; файлов
;****** глобальные константы
equ  имя1 = xxxx ;
equ  имя2 = nnnn
```

```
;****** глобальные регистровые переменные
def  имя1 = регистр
def  имя2 = регистр
;******* сегмент данных
.dseg
.org xxx  ; 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 xxx  ; адрес первого зарезервированного байта
.db выражение1,выражение2,.. ;записать список байтов в EEPROM
.dw выражение1,выражение2,... ;записать список слов в EEPROM
;****** сегмент кодов
.cseg
.org $000 ; адрес начала программы в программной памяти
.****** вектора прерываний (если они используются)
rjmp reset ; прерывание по сбросу
.org $001
rjmp INT0 ; обработчик внешнего прерывания 0
.org $002
rjmp INT1 ; обработчик внешнего прерывания 1
.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.
```

В комплексе предусмотрена работа с однофайловыми программами для учебных целей, поэтому подключение внешних файлов, кроме `.include "m8535def.inc"`, смысл имеет только для внешних сред программирования типа AVRStudio.

При использовании подпрограмм нужно обязательно определять стек. Для этого в начале основной программы нужно занести значения адреса вершины стека в регистры SPH и SPL.

4.4. Примеры программ

Ниже приводятся 3 программы для решения одной и той же простейшей задачи, демонстрирующие использование директив ассемблера.

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

Рис. 8. Алгоритм программы № 1

Задача следующая: вычесть из числа 5 число 3. Если включен тумблер на входе PA2, то на индикацию выдать результат вычитания. Если тумблер отключен – на индикацию вывести цифру ноль.

При работе с портами ввода/вывода следует учитывать, что направление передачи данных через отдельные выводы задается с помощью регистров DDR (DDRA, DDRB, DDRC, DDRD). Если вывод порта сконфигурирован как выход, то его переключение производится через регистр PORT (PORTA, PORTB, PORTC, PORTD), если вывод сконфигурирован как вход, то его опрос следует производить через регистр входных данных PIN (PINA, PINB, PINC, PIND).

Алгоритм программы (рис. 8) соответствует программе № 1, использующей директиву equ ассемблера.

```
     ;Программа №1 .Использование директивы equ
     .include "m8535def.inc" ;включить файл-описание ATmega8535
     .dseg           ;сегмент данных
     .equ cod0=$64   ;присвоение имен ячейкам ОЗУ
     .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           ;сегмент кодов
     .org 0          ;адрес начала программной памяти
     rjmp reset      ;вектор сброса
     .org $30        ;начало программы
reset: ldi r16,$00   ;определение стека с вершиной по адресу $00ff
     out sph,r16
     ldi r16,$ff
     out spl,r16
     ldi zl,$64     ;задание адреса начала зарезервированных ячеек
     ldi zh,$00

     ldi r16,$ff     ;настроить порт C на выход
     out ddrc,r16
     ldi r16,00      ;настроить порт A на вход
     out ddra,r16
     ldi r16,$ff     ;настроить порт B на выход
     out ddrb,r16
     ldi r16,$f0     ;настроить порт D: биты 0…3 на вход,
     out ddrd,r16    ;остальные на выход
     sbi portd,7     ;выдать 1 на разряд 7 порта D

     ldi r17,$3f     ;задание семисегментных кодов
     sts cod0,r17
     ldi r17,$06
     sts cod1,r17
```

```
      ldi r17,$5b
      sts cod2,r17
      ldi r17,$4f
      sts cod3,r17
      ldi r17,$66
      sts cod4,r17
      ldi r17,$6d
      sts cod5,r17
      ldi r17,$7d
      sts cod6,r17
      ldi  r17,$07
      sts  cod7,r17
      ldi  r17,$7f
      sts  cod8,r17
      ldi  r17,$6f
      sts  cod9,r17

      ldi  r17,5     ;задание уменьшаемого
      ldi  r18,3     ;задание вычитаемого

m1:   sbis pina,2    ;если включен тумблер, то пропустить
      rjmp m2        ;следующую команду
      mov  r20,r17   ;в r20 поместить уменьшаемое
      sub  r20,r18   ;вычесть вычитаемое
      rjmp vv
m2:   ldi  r20,0

vv:   push zl        ;сохранить zl в стеке
      add  zl,r20    ;сложить zl с результатом
      ld   r0,z      ;семисегментный код результата переслать в r0
      pop  zl        ;извлечь z1 из стека
      out  portc,r0  ;выдать результат на индикацию
      rjmp m1
```

Программа № 2 отличается от программы № 1 резервированием по одному байту оперативной памяти под семисегментные коды цифр от 0 до 9.

```
    ;Программа №2. Использование оперативной памяти под переменные
      .include "m8535def.inc"  ;подключение описания ATmega8535
      .dseg                    ;сегмент данных
      .org $64       ;адрес первого зарезервированного байта
      cod0:.byte 1   ;резервирование по одному байту
      cod1:.byte 1    ;под переменные
      cod2:.byte 1
      cod3:.byte 1
      cod4:.byte 1
      cod5:.byte 1
      cod6:.byte 1
      cod7:.byte 1
      cod8:.byte 1
      cod9:.byte 1
```

```
     .cseg          ;сегмент кодов
     .org $0        ;адрес начала программной памяти
     rjmp reset     ;вектор сброса
reset: ldi r16,$00 ;определение стека с вершиной по адресу $00ff
     out  sph,r16
     ldi  r16,$ff
     out  spl,r16
     ldi  zl,$64   ;задание адреса начала зарезервированных ячеек
     ldi  zh,$00

     ldi  r16,$ff  ;настроить порт C на выход
     out  ddrc,r16
     ldi  r16,00   ;настроить порт A на вход
     out  ddra,r16
     ldi  r16,$ff  ;настроить порт B на выход
     out  ddrb,r16
     ldi  r16,$f0  ;настроить порт D: биты 0…3 на вход,
     out  ddrd,r16 ;остальные на выход
     sbi  portd,7  ;выдать 1 на разряд 7 порта D

     ldi  r17,$3f  ;задание семисегментных кодов
     sts  cod0,r17
     ldi  r17,$06
     sts  codl,r17
     ldi  r17,$5b
     sts  cod2,rl7
     ldi  r07,$4f
     sts  cod3,rl7
     ldi  r17,$66
     sts  cod4,rl7
     ldi  r17,S6d
     sts  cod5,r17
     ldi  r17,$7d
     sts  cod6,r17
     ldi  r17,$07
     sts  cod7,r17
     ldi  r17,$7f
     sts  cod8,r17
     ldi  rl7,$6f
     sts  cod9,r17

     ldi  r17,5    ;задание уменьшаемого
     ldi  r18,3    ;задание вычитаемого

m1:  sbis pina,2   ;если включен тумблер, то пропустить
     rjmp m2       ;следующую команду
     mov  r20,r17  ;в r20 поместить уменьшаемое
     sub  r20,r18  ;вычесть вычитаемое
     rjmp vv
m2:  ldi r20,0

vv:  push zl       ;сохранить zl в стеке
     add z1,r20    ;сложить zl с результатом
```

```
     ld  r0,z      ;семисегментный код результата переслать в r0
     pop z1        ;извлечь zl из стека
     out portc,r0  ;выдать результат на индикацию
     rjmp m1
```

Программа № 3 самая короткая. Она использует директиву .dw для определения слов в программной памяти. В программе используется команда LPM ассемблера. По этой команде загружается байт, адресуемый регистром Z в регистр R0. Команда обеспечивает доступ к любому байту памяти программы, организованной как 16-битное слово. Младший бит регистра Z определяет, осуществляется ли доступ к младшему байту слова (0) или к старшему (1).

```
     ;Программа №3. Использование директивы .dw
     .include "m8535def.inc" ;подключение описания ATmega8535
     .cseg
     .org 0
     rjmp reset
     .dw  $063f,$4f5b,$6d66,$077d,$617f,$7c77,$5e39,$7179
     ;семисегментные коды цифр от 0 до F
reset: ldi r16,$00  ;определение стека с вершиной по адресу $00ff
     out  sph,r16
     ldi  r16,$ff
     out  spl,r16

     ldi  rl6,$ff   ;настроить порт C на выход
     out  ddrc,r16
     ldi  r16,00    ;настроить порт A на вход
     out  ddra,r16
     ldi  r16,$ff   ;настроить порт B на выход
     out  ddrb,r16
     ldi  r16,$f0   ;настроить порт D: биты 0...3 на вход,
     out  ddrd,r16  ;остальные на выход
     sbi  portd,7   ;выдать 1 на разряд 7 порта D

     ldi  zl,02     ;установить адрес семисегментного кода нуля
     ldi  zh,00     ;в регистр Z
     ldi  r17,5     ;задание уменьшаемого
     ldi  r18,3     ;задание вычитаемого

m1:  sbis pina,2    ;если включен тумблер, то пропустить
     rjmp m2        ;следующую команду
     mov r20,r17    ;в r20 поместить уменьшаемое
     sub r20,r18    ;вычесть вычитаемое
     rjmp vv
m2:  ldi r20,0      ;присвоить результату значение 0

vv:  push z1        ;сохранить z1 в стеке
     add z1,r20     ;сложить z1 с результатом
     lpm       ;загрузить байт, адресуемый регистром Z, в регистр 0
     pop z1         ;извлечь z1 из стека
     out portc,r0   ;выдать результат на индикацию
     rjmp m1
```

5. ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ ПРОГРАММ AVR-STUDIO

5.1. Возможности программного пакета AVR-Studio

Фирмой «Atmel» разработан программный пакет поддержки разработок на AVR-микроконтроллерах в среде Windows – AVR-Studio. AVR-Studio – это интегрированная отладочная среда разработки программ (Integrated Development Environment – IDE), включающая транслятор языка ассемблера AVR-микроконтроллеров, отладчик и программное обеспечение верхнего уровня для поддержки внутрисхемного программирования. Отладчик AVR-Studio поддерживает все типы микроконтроллеров AVR и имеет два режима работы: режим программной симуляции и режим управления различными типами внутрисхемных эмуляторов производства фирмы «Atmel». Отладочная среда поддерживает выполнение программ как в виде ассемблерного текста, так и в виде исходного текста языка С. AVR-Studio распространяется свободно, его последняя версия всегда доступна на сайте фирмы «Atmel».

Студентам при подготовке к лабораторным занятиям и предварительной отладке программ, предназначенных для лабораторного комплекса, рекомендуется использовать пакет AVR-Studio.

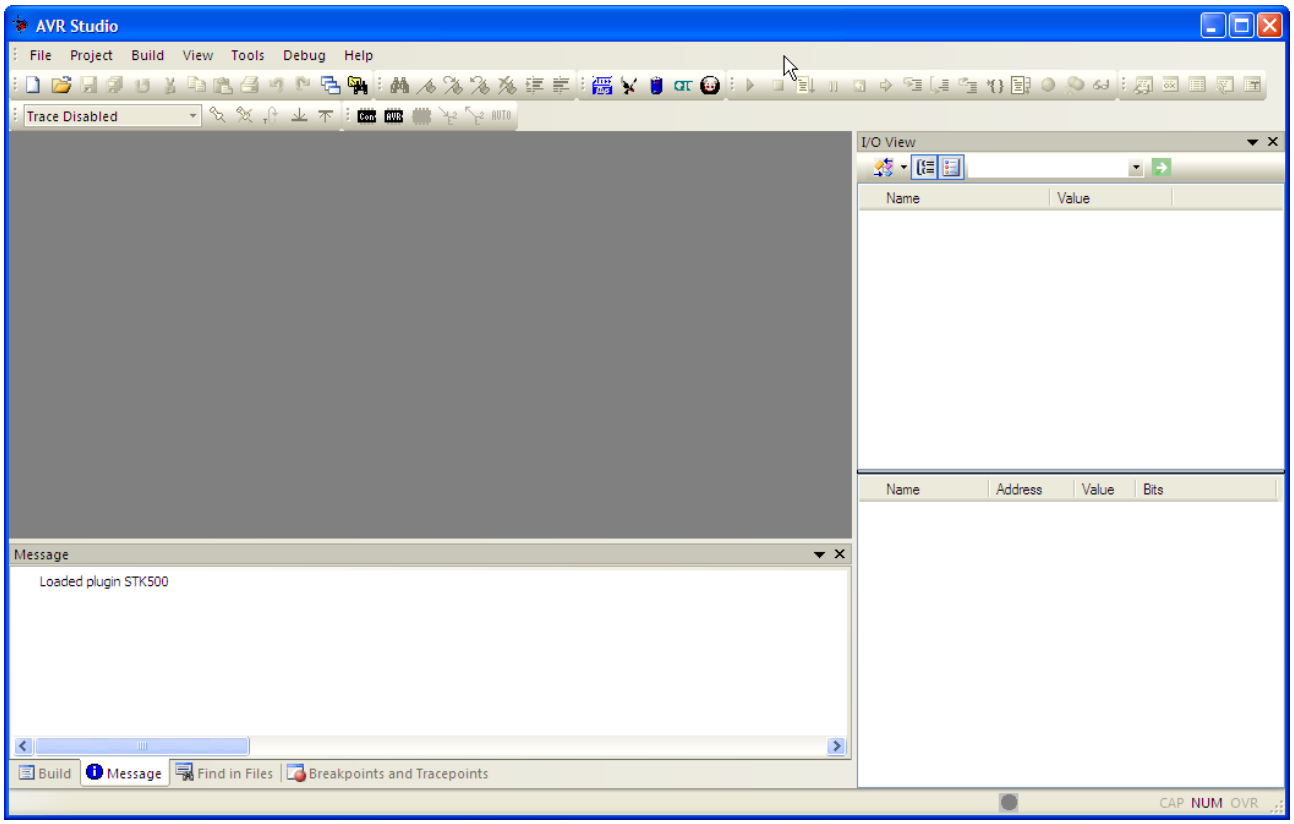
Лабораторный комплекс можно также использовать и для решения реальных задач (производственных), требующих подготовки и записи программы в микроконтроллер ATMega8535, используемый в качестве встраиваемого контроллера. В этом случае используется одно рабочее место (остальные рабочие места могут быть отключены). Можно отказаться от работы монитора ПЭВМ в восьмиоконном режиме и от ввода программы с клавиатуры рабочего места. Ввод программы в ПЭВМ и её редактирование удобнее осуществлять с клавиатуры преподавателя. Модуль рабочего места используется для записи программы в микроконтроллер.

Для создания и отладки программы при работе с одним рабочим местом также целесообразно использовать программу AVR-Studio. Она позволяет в том числе и записывать программу в микроконтроллер, но в стенде используется нестандартная схема программатора, поэтому использовать эту возможность программы нельзя.

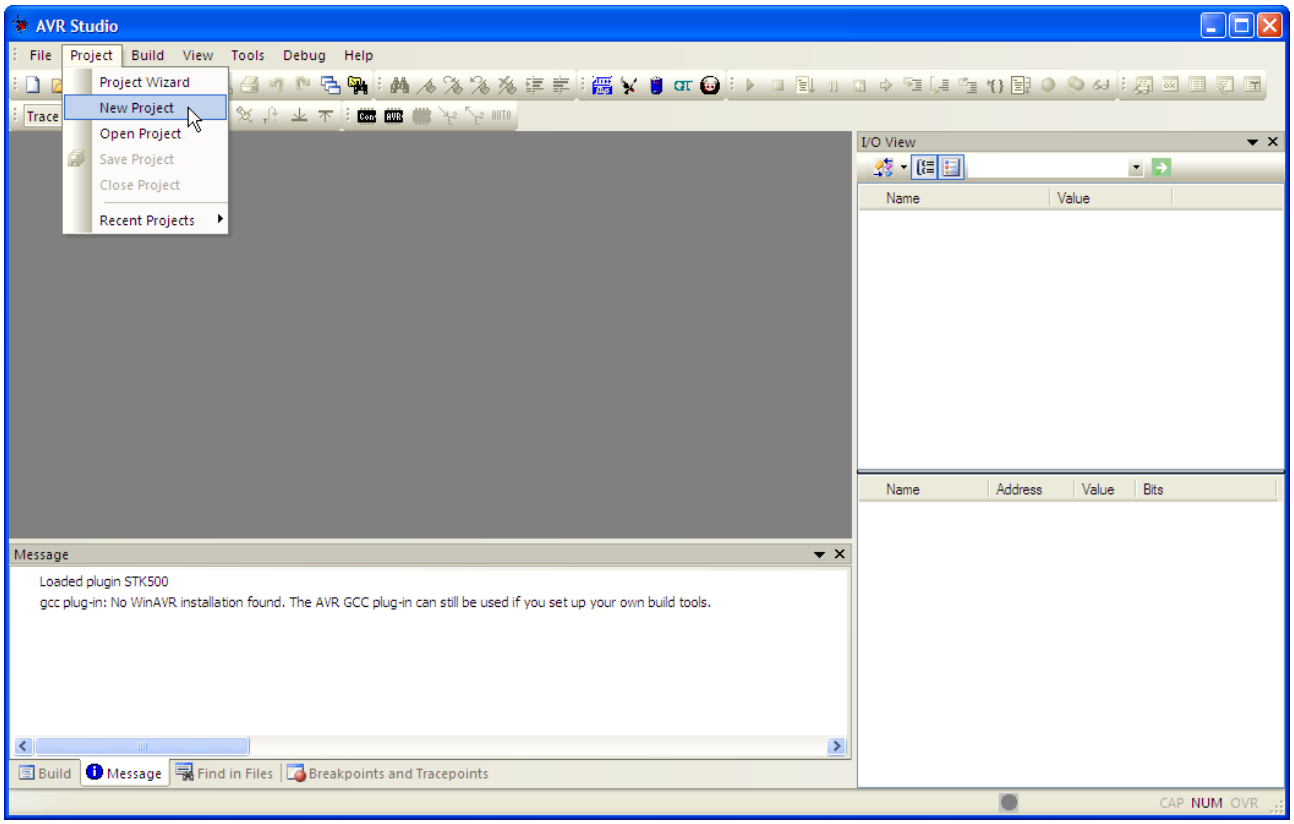
Для записи программы в микроконтроллер используется программа какого-либо рабочего места, в которую надо загрузить текст программы на языке ассемблера, после чего нужно нажать кнопку прошивки. Произойдет компиляция, и если ошибок нет, прошивка микроконтроллера на рабочем месте.

5.2. Создание и компиляция программы в AVR-Studio

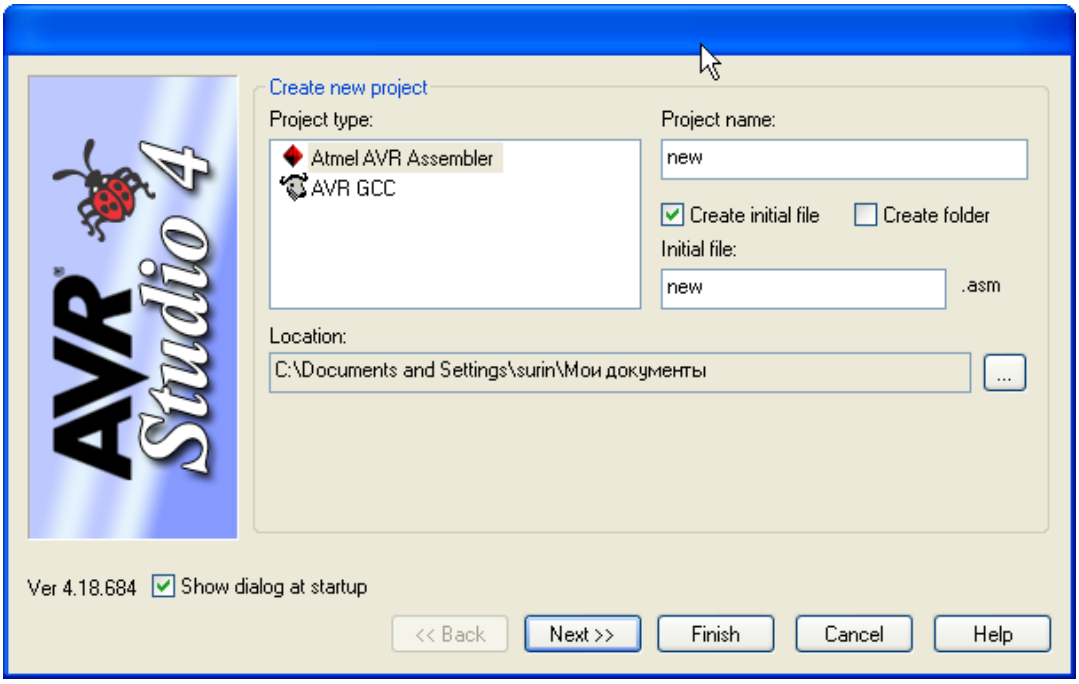
Для запуска программы запустите файл AvrStudio.exe. Появится основное диалоговое окно программы.



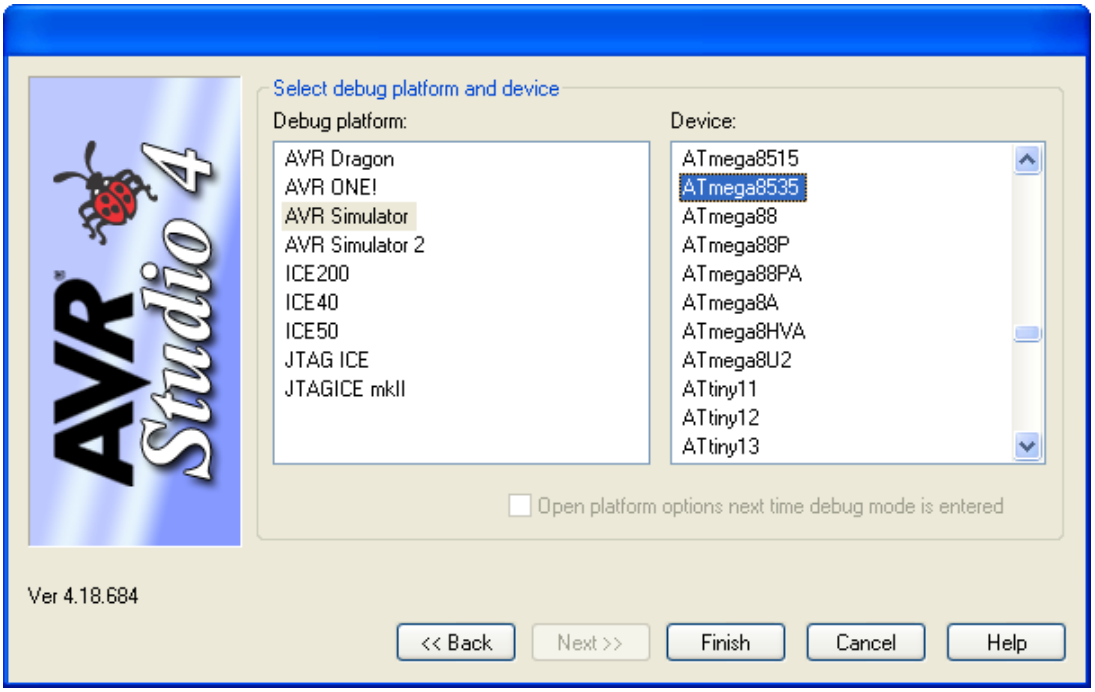
В верхней части программы находится меню, в нем нужно выбрать Project → New Project.



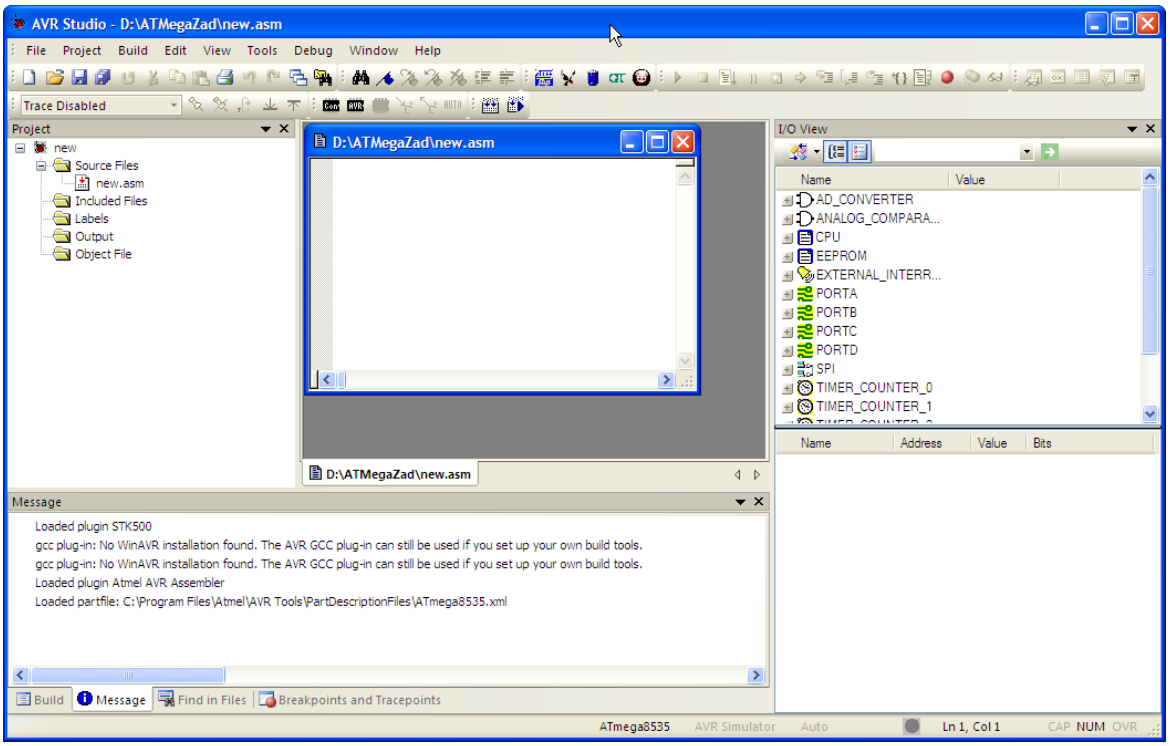
В появившемся окне выберите имя проекта (Project name), место на диске, куда сохранять проект (Location), а также тип проекта (Project type), щелкнув мышью на AVR assembler, затем щелкнуть Next >>.



Появится окно выбора платформы отладки и устройства (Select debug platform and device). В пункте Debug platform необходимо выбрать AVR Simulator, в пункте Device выбрать Atmega8535, затем щелкнуть на кнопку Finish.



В появившемся окне Project щелкаем два раза на asm файле и в открывшемся окне набираем программу.



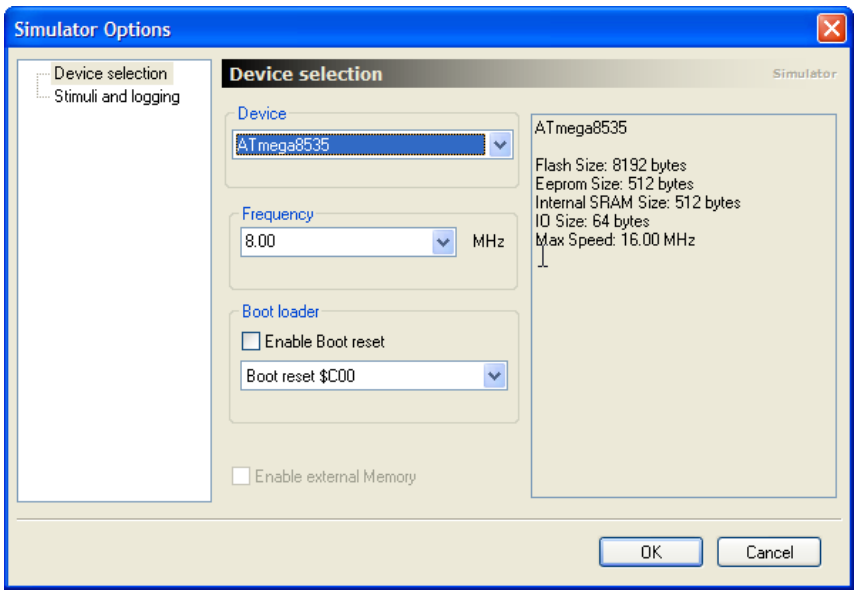
После того как программа набрана, в верхнем меню выбираем Build и производим ее компиляцию, при этом создается файл с расширением hex, который затем надо будет записать в микроконтроллер. После компиляции появится окно Build, в котором указано, какой файл ассемблируется, используемый файл библиотеки, количество слов в программе и сообщение об отсутствии ошибок Assembly complete with 0 errors, 0 warnings. Если есть ошибки, то в этом окне указываются тип ошибки, номер строки с ошибкой и в конце общее число ошибок. Для их исправления необходимо вернутся к редактируемому файлу, а затем снова откомпилировать программу.

5.3. Отладка программы в AVR-Studio

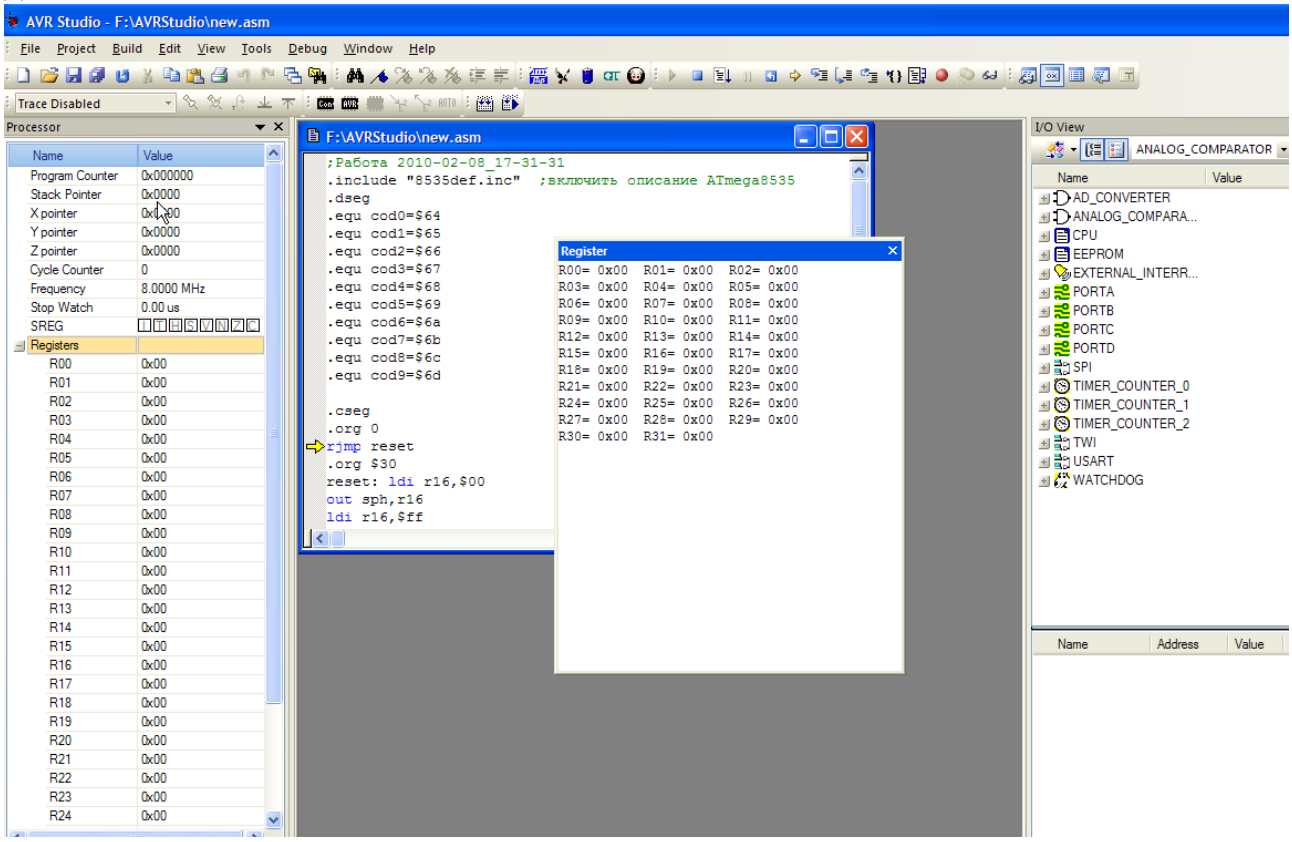
AVR-Studio позволяет не только компилировать программы, но и отлаживать их на этапе разработки. При этом AVR-Studio эмулирует работу микроконтроллера, всех портов ввода/вывода, счетчиков/таймеров, прерываний, ШИМ и АЦП. Эмуляция работы программы позволяет рассмотреть ее работу, как если бы она была записана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

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

Для отладки программы, после того как она написана, нужно в меню Build выбрать пункт Build and run. Вызвать окно опций эмулятора (Simulation Options) в меню Debug → AVR Simulator Options. В пункте устройство (Device) нужно выбрать микроконтроллер ATmega8535, в пункте частота (Frequency) частоту 8 МГц, нажать кнопку ОК.



После этого появится окно, в котором набиралась программа, но начало программы будет отмечено желтой стрелкой — это начало программы, выше идут директивы компилятора. При эмуляции работы программы необходимо видеть состояния регистров, портов ввода/вывода, процессора. Для просмотра регистров в главном меню программы выбираем пункт просмотр (View), затем пункт регистры (Registers), для просмотра состояния процессорного ядра используется панель процессор (View → Toolbars → Processor), порты ввода/вывода и периферийные модули удобно наблюдать через панель ввода/вывода (View → Toolbars → I/O). В меню View имеются и другие пункты, которые можно использовать, такие как память (Memory) для просмотра памяти данных и программ. Таким образом, можно получить окно примерно такого вида:



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 выполнятся 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.

На панели ввода/вывода I/O View, где 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 содержание какого-нибудь регистра, можно изменить его состояние в процессе выполнения программы. В регистрах портов ввода/вывода можно задать входные сигналы, отмечая галочкой в нужном бите состояния логической единицы, тем самым эмулируется воздействие внешних сигналов.

Прочие возможности AVR-Studio могут быть изучены пользователем в процессе работы с программой.

6. ЛАБОРАТОРНЫЕ РАБОТЫ ПО ИЗУЧЕНИЮ AVR-МИКРОКОНТРОЛЛЕРОВ

6.1. Общие положения

Ниже предлагаются методические указания к лабораторным работам, позволяющим изучить основные функциональные возможности программируемых микроконтроллеров типа ATmega8535 семейства AVR и приобрести навыки их программирования.

К лабораторным работам в сжатой форме даны пояснения, позволяющие при знании общих принципов функционирования изучаемых узлов микроконтроллера подготовиться к занятию без привлечения дополнительной литературы. В каждой из них представлено 16 вариантов индивидуальных заданий (для двух подгрупп по 8 рабочих мест). По своему усмотрению преподаватель может упростить или усложнить задачу варианта в зависимости от подготовки обучающегося. Все лабораторные работы рассчитаны на несколько часов самостоятельной работы при домашней подготовке и оформлении отчета.

Содержание работ:

1. Ознакомиться с описанием лабораторной работы и индивидуальным заданием, изучить теоретический материал.

2. Дома, при подготовке к работе, для индивидуального задания составить схему алгоритма его решения и написать программу на языке ассемблера (рекомендуется предварительно отладить программу с помощью AVR-Studio).

3. В лаборатории со своего рабочего места ввести подготовленную программу в ПЭВМ и осуществить ее компиляцию (можно загрузить с компьютера преподавателя подготовленный в AVR-Studio файл с ассемблерным текстом программы) .

4. При отсутствии ошибок в программе записать ее через программатор в ПЗУ программируемого микроконтроллера.

5. На рабочем месте программируемый микроконтроллер включается для выполнения заданной задачи. Путем подачи на входы микроконтроллера необходимых входных сигналов и визуального наблюдения выходных сигналов, выдаваемых микроконтроллером, оценивается правильность работы подготовленной программы.

Отчет по лабораторной работе должен содержать:

а) цель работы;

б) условие индивидуального задания;

в) адресацию входных и выходных переменных;

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

д) программу на языке ассемблера или листинг программы;

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

ж) выводы по работе.

6.2. Лабораторная работа № 1. Изучение системы команд микроконтроллера и системы параллельного ввода/вывода

Цель работы

Ознакомиться с лабораторным комплексом, изучить систему команд микроконтроллера ATmega8535 и организацию 8-разрядных двунаправленных портов для ввода и вывода дискретной информации.

Пояснения к работе

Для выполнения работы необходимо знать структуру и функционирование микроконтроллера ATmega8535, способы адресации и систему команд. Система команд микроконтроллеров семейства AVR представлена в разделе 2.

Микроконтроллер ATmega8535 имеет 4 параллельных восьмиразрядных порта ввода/вывода A, B, C и D.

Взаимодействие с каждым портом осуществляется через три регистра в пространстве ввода/вывода памяти данных: регистр данных порта PORT, регистр направления данных DDR и регистр выводов порта PIN. Например, для порта A регистр данных обозначается PORTA и имеет адрес ввода/вывода $1B, если же к нему обращаться как к ячейке памяти, то адрес будет иметь значение $3B. Регистр направления данных порта A обозначается DDRA, его адрес – $1A($3A), регистр выводов – PINA, адрес – $19($39).

Регистр данных порта A – PORTA

Бит	7	6	5	4	3	2	1	0	
$1B($3B)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
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	

Регистр направления данных порта A – DDRA

Бит	7	6	5	4	3	2	1	0	
$1A ($3A)	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0	DDRA
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	

Регистр входных данных порта A – PINA

Бит	7	6	5	4	3		1	0	
$19 ($39)	PINA7	PINA6	PINA5	P1NA4	PINA3	PINA2	PINA1	PINA0	PINA
Чтение/Запись	R	R	R	R	R	R	R	R	
Исходное значение	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Регистр PINA обеспечивает только возможность чтения, а регистры PORTA и DDRA – возможность чтения и записи. Регистр PINA не является регистром в полном смысле этого слова. Обращение к нему обеспечивает чтение физического состояния каждого вывода порта.

Аналогичное устройство и обозначение имеют и регистры портов B, C и D, их адреса приведены в таблице 9.

Таблица 9. Адреса регистров портов ввода/вывода B, C и D

Регистр	PORTB	DDRB	PINB	PORTC	DDRC	PINC	PORTD	DDRD	PIND
Адрес	$18($38)	$17($37)	$16($36)	$15($35)	$14($34)	$13($33)	$12($32)	$11($31)	$10($30)

Все порты в качестве цифровых портов ввода/вывода общего назначения работают одинаково:

- Каждый вывод порта в любой момент времени. может быть запрограммирован индивидуально на ввод или вывод. Если в регистре направления данных порта DDR для рассматриваемого бита записать “0”, то соответствующий вывод конфигурируется как вход, при записи “1” – как выход.
- Биты регистра PORT выполняют двойную функцию. Если вывод функционирует как выход, то этот бит определяет состояние вывода порта. Если бит установлен в 1, на выводе устанавливается напряжение высокого уровня (4,5…5 В). Если бит сброшен в 0, на выводе устанавливается напряжение низкого уровня (0…0,5 В). Нагрузочной способности каждого вывода достаточно для непосредственного управления светодиодным индикатором.
- Если вывод функционирует как вход, то бит регистра PORT определяет состояние внутреннего подтягивающего резистора для данного вывода. При установке бита в 1 подтягивающий резистор подключается между выводом микроконтроллера и линией питания (уровень напряжения на выводе неподключенного входа составляет 3,5…4 В, что воспринимается как наличие входного сигнала “1”, но при этом велико влияние помех).
- Все выводы портов незапрограммированного микроконтроллера находятся в третьем состоянии.

Контакты ввода/вывода AVR-микроконтроллеров могут иметь дополнительные функции и использоваться различными периферийными модулями. При этом возможны две ситуации. В одних случаях пользователь должен самостоятельно задавать конфигурацию вывода, а в других вывод конфигурируется автоматически при включении соответствующего периферийного устройства. Об этом сказано при рассмотрении соответствующих периферийных устройств.

Порт A в микроконтроллере ATmega8535 служит также для ввода аналоговых сигналов в аналого-цифровой преобразователь. Выводы порта B могут выполнять альтернативные функции, указанные в таблице 10 (при этом регистры PORTB, DDRB должны быть установлены соответствующим образом).

Таблица 10. Альтернативные функции выводов порта B

Вывод	Альтернативная функция
PB0	T0 – вход тактового сигнала таймера/счетчика 0
PB1	T1 – вход тактового сигнала таймера/счетчика 1
PB2	AIN0 – положит. вывод компаратора / INT2 – вход внешнего прерывания 2
PB3	AIN1 – отрицат. вывод компаратора / OC0 – вывод сравнения выхода таймера/счетчика 0
PB4	SS – вход выбора ведомого SPI
PB5	MOSI – установка ведущий выход/ведомый вход SPI
PB6	MISO – установка ведущий вход/ведомый выход SPI
PB7	SCK – тактовый сигнал SPI

У порта C четыре вывода могут выполнять альтернативные функции: выводы PC6 и PC7 выполняют функции TOSC1 и TOSC2 таймера/счетчика 2 (вход и выход для подключения резонатора), выводы PC0 и PC1 выполняют функции SCL (синхронизация) и SDA (данные) для последовательного двухпроводного интерфейса TWI. Выводы порта D могут выполнять альтернативные функции, указанные в таблице 11 (регистры PORTD, DDRD должны быть установлены соответствующим образом).

Таблица 11. Альтернативные функцин выводов порта D

Вывод	Альтернативная функция
PD0	RxD – вход приемника USART
PD1	TxD – выход передатчика USART
PD2	INT0 – вход внешнего прерывания 0
PD3	INT1 – вход внешнего прерывания 1
PD4	OC1B – вывод сравнения выхода B таймера/счетчика 1
PD5	OC1A – вывод сравнения выхода A таймера/счетчика 1
PD6	ICP – вход триггера захвата таймера/счетчика 1
PD7	OC2 – вывод сравнения выхода таймера/счетчика 2

При разработке программ в соответствии с полученными индивидуальными заданиями студенты могут взять за основу примеры программ, приведенных в разделе 4.4. Для выполнения некоторых заданий необходимо обеспечивать временные задержки, которые можно организовать при помощи следующей подпрограммы:

```
wait:               ; подпрограмма временной задержки
     ldi  r21,$28
loop2:
     ldi  r22,$0ff
loop1:
     ldi  r23,$0ff
loop:
     dec  r23
     brne  loop
     dec  r22
     brne  loop1
     dec  r21
     brne  loop2
     ret
```

Приведенная подпрограмма, использующая регистры R21 – R23, обеспечивает при тактовой частоте 8 МГц временную задержку примерно в 1 секунду. При необходимости изменения задержки можно изменить число, записываемое в регистр R21.

Варианты индивидуальных заданий

1. Организовать сложение двух трехбитных чисел и индикацию результата на семисегментном индикаторе. Три произвольно выбранных тумблера представляют собой первое слагаемое, т.е. возможен набор чисел от 0 до 7. Три

других тумблера представляют собой второе слагаемое. На индикаторе высвечивать результат в шестнадцатеричном формате, т.е. 0 ... E.

2. В памяти записать массив, содержащий 8 ячеек. В ячейках массива находятся шестнадцатеричные цифры (числа от 00 до 0F). С помощью четырех тумблеров вводится еще одна шестнадцатеричная цифра. Программа должна определить, присутствует ли эта цифра в массиве. Если цифра присутствует, то нужно высветить ее в шестнадцатеричном виде на семисегментном индикаторе. Если цифры в массиве нет, на индикаторе высвечивается символ "H" (нет).

3. При нажатии на одну кнопку загорается цифра 1 и светодиод VD4. При нажатии на другую кнопку загорается цифра 2 и светодиод VD5. При одновременном нажатии этих кнопок загорается цифра 3 и включается светодиод VD6.

4. Организовать умножение двухбитных чисел и индикацию результата на семисегментном индикаторе. Два произвольно выбранных тумблера представляют собой первый сомножитель, т.е. возможен набор чисел от 0 до 3. Два других тумблера представляют собой второй сомножитель.

5. При отключенном тумблере горит светодиод VD4 и на индикаторе HG1 горит цифра 2. При включении тумблера светодиод VD4 и индикатор HG1 гаснут, загорается светодиод VD5 и на индикаторе HG2 загорается цифра 5. При включении другого тумблера все светодиоды и индикаторы гаснут.

6. Организовать сложение двух чисел 3 и 4. При нажатой кнопке на семисегментном индикаторе горит первое слагаемое, при другой нажатой кнопке – второе слагаемое, при третьей нажатой кнопке – результат.

7. В массиве из 16 ячеек памяти располагаются шестнадцатеричные числа от 00 до 0F. В массиве есть только одно число, которое повторяется несколько раз. Выявить какое это число и сколько раз оно повторяется. При нажатии на кнопку на семисегментном индикаторе HG1 загорается повторяющееся число. При нажатии на другую кнопку на семисегментном индикаторе HG2 высвечивается число его повторений. Если нет повторяющегося числа, то при нажатии на первую кнопку на индикаторе HG1 загорается символ "H" (нет).

8. При включении микроконтроллера на семисегментном индикаторе горит "0". Организовать счет и индикацию числа нажатий кнопки на индикаторе 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 сопровождается загоранием светодиода VD5. Для упрощения программы мер борьбы с дребезгом контактов можно не предпринимать.

9. В исходном состоянии на семисегментном индикаторе горит цифра 7. При нажатии и отпускании кнопки на индикаторе загорается цифра 5 и включаются светодиоды VD4 и VD6. При нажатии и удержании другой кнопки все индикаторы и светодиоды гаснут, при отпускании возобновляется исходное состояние.

10. В исходном состоянии горят светодиоды VD4 и VD6, и на семисегментном индикаторе горит цифра 4. При нажатии и удержании кнопки светодиоды VD4 и VD6 гаснут, загорается светодиод VD5, а на индикаторе загорается цифра 9. При отпускании кнопки схема приходит в исходное состояние.

11. В исходном состоянии горит светодиод VD1, а на семисегментном индикаторе высвечивается цифра 1, то есть номер светодиода. При нажатии и отпускании кнопки светодиод VD1 гаснет, а VD2 загорается, то есть происходит сдвиг свечения влево. На индикаторе загорается цифра 2. При каждом очередном нажатии на кнопку свечение сдвигается влево, то есть наблюдается свечение VD3, VD4, VD5, VD6, VD7, VD8, затем снова VD1 и так далее. При этом на индикаторах высвечиваются соответственно цифры 3, 4, 5, 6, 7, 8, 1 и так далее. При нажатии и отпускании другой кнопки схема работает аналогично, но сдвиг свечения происходит вправо.

12. Реализовать на микроконтроллере схему управления светофором. При включении тумблера светофор работает в дневном режиме, то есть чередование сигналов следующее: зеленый (VD1), желтый (VD2), красный (VD3), желтый, зеленый, желтый и так далее. При отключении тумблера светофор работает в ночном режиме, то есть мигает желтый светодиод VD2. В дневном режиме работы на семисегментном индикаторе горит буква “d”, в ночном режиме – буква “H”.

13. Организовать «бегущий» огонь по сегментам семисегментного индикатора HG1 и HG2. При «беге» по часовой стрелке чередование сегментов следующее: a, b, c, d, e, f, a, b … и так далее. При включенном тумблере реализуется «бегущий» огонь по часовой стрелке, при отключенном – против часовой стрелки. При другом включенном тумблере «бегущий» огонь реализуется по сегментам индикатора HG1, при отключенном – по сегментам индикатора HG2.

14. Организовать счет числа нажатий двух кнопок. В исходном состоянии на семисегментном индикаторе горит число 0. При каждом очередном нажатии на одну кнопку число на индикаторе увеличивается на единицу. Счет возможен до F. Если счет достиг числа F, то дальнейшие нажатия этой кнопки число не меняют. При каждом очередном нажатии на другую кнопку число на индикаторах уменьшается на единицу. При достижении числа 0 дальнейшие нажатия этой кнопки число не меняют.

15. Организовать счет числа нажатий кнопки. В исходном состоянии все светодиоды и индикаторы погашены. Каждое очередное нажатие кнопки считается микроконтроллером. После 10 нажатий загорается светодиод VD4, а на семисегментном индикаторе загорается цифра 1. После второго десятка нажатий дополнительно загорается светодиод VD5, а на индикаторе цифра 2. Дальнейшие нажатия кнопки не меняют состояния схемы. Нажатие другой кнопки гасит все светодиоды и индикатор, и микроконтроллер приходит в исходное состояние.

16. Реализовать последовательное формирование свечения цифры 3 на семисегментном индикаторе. В исходном состоянии индикатор погашен. При включении тумблера включаются сегменты индикатора HG1 в последовательности a, b, c, d, g. На индикаторе горит цифра 3. При отключении тумблера гаснут сегменты индикатора HG1 в последовательности g, d, c, b, a.

6.3. Лабораторная работа № 2. Система внешних прерываний микроконтроллера ATmega8535 семейства AVR

Цель работы

Изучить организацию прерываний в микроконтроллере ATmega8535 и обслуживание подсистемы внешних прерываний.

Пояснения к работе

Микроконтроллер ATmega8535 использует 21 источник 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 состояния SREG.

Перечень векторов прерывания представлен в таблице 12.

Таблица 12. Вектора прерываний микроконтроллера ATMega8535

№ вектора	Адрес	Источник	Примечание
1	$000	RESET	Сброс по выводу RESET и сторожевому таймеру
2	$001	INT0	Запрос внешнего прерывания 0
3	$002	INT1	Запрос внешнего прерывания 1
4	$003	TIMER2 COMP	Совпадение при сравнении таймера/счетчика 2
5	$004	TIMER2 OVF	Переполнение таймера/счетчика 2
6	$005	TIMER1 CAPT	Захват таймера/счетчика 1
7	$006	TIMER1 COMPA	Совпадение A при сравнении таймера/счетчика 1
8	$007	TIMER1 COMPB	Совпадение B при сравнении таймера/счетчика 1
9	$008	TIMER1 OVF	Переполнение таймера/счетчика 1
10	$009	TIMER0 OVF	Переполнение таймера/счетчика 0
11	$00A	SPI, STC	Завершение пересылки SPI
12	$00B	USART, RXC	Завершение приема USART
13	$00C	USART, UDRE	Регистр данных USART пуст
14	$00D	USART, TX	Завершение передачи USART
15	$00E	ADC	Завершение аналого-цифрового преобразования
16	$00F	EE_RDY	Готовность EEPROM
17	$010	ANA_COMP	Срабатывание аналогового компаратора
18	$011	TWI	Последовательный двухпроводной интерфейс
19	$012	INT2	Внешнее прерывание 2
20	$013	TIMER0 COMP	Совпадение при сравнении таймера/счетчика 0
21	$014	SPM_RDY	Готовность SPM

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

При возникновении прерывания бит I разрешения глобального прерывания очищается, и все прочие прерывания запрещаются. Пользовательское программное обеспечение, с тем, чтобы разрешить вложенные прерывания, может установить бит I внутри подпрограммы обработки прерывания. Выход из подпрограммы обработки прерывания происходит по команде RETI, при этом бит I устанавливается в состояние 1.

Все имеющиеся прерывания можно разделить на два типа. Прерывания первого типа генерируются при наступлении некоторого события, в результате которого устанавливается флаг прерывания. Затем, если прерывание разрешено, в счетчик команд загружается адрес вектора соответствующего прерывания. При этом флаг прерывания аппаратно сбрасывается. Он также может быть сброшен программно, записью логической единицы в бит регистра, соответствующий флагу.

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.

Прерывания второго типа не имеют флагов прерываний и генерируются в течение всего времени, пока присутствуют условия, необходимые для генерации прерывания. Например, внешние прерывания по уровню сигнала флага не имеют и условия прерывания сохраняются, пока активен внешний сигнал. Соответственно если условия, вызывающие прерывание, исчезнут до разрешения прерывания, генерации прерывания не произойдет.

Необходимо иметь в виду, что регистр состояния автоматически не сохраняется при переходе к прерывающей подпрограмме, т.е. при необходимости он должен быть программно сохранён и при возвращении из прерывающей подпрограммы восстановлен.

Микроконтроллер ATmega8535 содержит общий регистр управления прерываниями GICR и общий регистр флагов внешних прерываний GIFR.

Общий регистр управления прерываниями – GICR

Бит	7	6	5	4	3	2	1	0	
$3B($5B)	INT1	INT0	INT2	-	-	-	INSEL	IVCE	**GICR**
Чтение/Запись	R/W	R/W	R/W	R	R	R	R/W	R/W	
Исходное значение	0	0	0	0	0	0	0	0	

- Биты 7, 6 и 5 – **INT1, INT0, INT2: Разрешение запроса внешнего прерывания 1, 0, 2 (маска).** При установленных битах INT1, INT0, INT2 и установленном бите I регистра статуса (SREG) разрешаются прерывания по соответствующим выводам внешних прерываний. Активизация выводов INT1, INT0 и INT2 вызывает запрос прерывания, если даже эти выводы сконфигурированы как выходы.
- Биты 4 ... 2 – **Res: Reserved Bits – Зарезервированные биты.** Эти биты зарезервированы в ATmega8535 и при считывании всегда покажут состояние 0.
- Бит 1 – **INSEL: Положение таблицы векторов прерываний в памяти программ.** Если флаг сброшен в 0, то таблица векторов прерываний распо-

лагается в начале памяти программ (в соответствии с табл. 12), если установлен в 1 — в начале области загрузчика.

- Бит 0 – **IVCE: Разрешение изменения положения таблицы векторов прерываний.** Бит устанавливается программно и удерживается в состоянии 1 в течение 4 тактов, давая возможность изменения флага INSEL.

Общий регистр флагов внешних прерываний – GIFR

Бит	7	6	5	4	3	2	1	0	
$3A($5A)	INTF1	INTF0	INTF2	-	-	-	-	-	**GIFR**
Чтение/Запись	R/W	R/W	R/W	R	R	R	R	R	
Исходное значение	0	0	0	0	0	0	0	0	

- Биты 7, 6 и 5 – **INTF1, INTF0, INTF2: Флаги внешних прерываний INTF1, INTF0, INTF2.** В случае поступления запроса прерывания на какой-либо из указанных выводов устанавливается соответствующий флаг прерывания. Если бит I регистра SREG и соответствующий бит разрешения регистра GICR установлены, то выполняется переход по вектору прерывания. При возврате из процедуры прерывания флаг автоматически очищается. Кроме того, флаг можно очистить, записав в него логическую единицу. Флаг сброшен постоянно, если генерация прерывания должна происходить по низкому уровню на соответсвующем выводе.
- Биты 4 ... 0 – **Res: Reserved Bits** – **Зарезервированные биты.** Эти биты зарезервированы в ATmega8535 и при считывании показывают состояние 0.

Регистр управления MCUCR содержит служебные биты для общих функций микроконтроллера. Биты управления опознанием прерывания определяют условия возникновения запросов на входах INT0 и INT1: по нарастающему/спадающему фронту сигнала или по логическому уровню.

Регистр управления микроконтроллером - MCUCR

Бит	7	6	5	4	3	2	1	0	
$35($55)	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	**MCUCR**
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	

- Биты 7 ... 4 управляют режимами энергосбережения.
- Биты 3, 2 – **ISC11, ISC10: Interrupt Sense Control 1 bit 1 and bit 0** – **Биты управления идентификацией внешнего прерывания 1.**
- Биты 1, 0 – **ISC01, ISC00: Interrupt Sense Control 0 bit 1 and bit 0** – **Биты управления идентификацией внешнего прерывания 0.**

Биты ISC задают условия формирования запросов прерывания по входам INT1 и INT0 в соответствии с таблицей 13.

Таблица 13. Задание характера сигналов прерываний 1 и 0

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

При изменении битов ISC в регистре MCUCR соответствующее прерывание должно быть запрещено путем очистки бита разрешения прерывания в регистре GICR. В ином случае может произойти прерывание.

Условия возникновения запроса прерывания 2 задаются битом ISC2 в регистре управления и состояния микроконтроллера MCUCSR.

Регистр управления и состояния микроконтроллера – MCUCSR

Бит	7	6	5	4	3	2	1	0	
$34($54)	JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF	**MCUCSR**
Чтение/Запись	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Исходное значение	0	0	0	0	0	0	0	0	

- Бит 6 – **ISC2: Interrupt Sense Control 2 – Бит управления идентификацией внешнего прерывания 2.** Если этот бит сброшен в 0, то запрос прерывания генерируется по спадающему фронту сигнала на INT2, если бит установлен в 1, то запрос прерывания генерируется по нарастающему фронту сигнала на INT2. В отличие от входов внешних прерываний INT0 и INT1 на входе INT2 обнаружение фронтов сигналов происходит асинхронно, т. е. не требует наличия тактового сигнала.
- Остальные биты регистра MCUCSR используются для индикации состояния и управления различными программно-аппаратными средствами микроконтроллера.

При проведении лабораторной работы запускается фоновая программа, которая будет прерываться внешними источниками INT0 и INT1. Разрешение прерываний и условия возникновения сигналов запроса прерывания задаются с тумблеров блока управления.

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) повторить действия пунктов б) и в) при задании прерываний по уровню и проанализировать, какие отличия имеются при обработке прерываний по уровню.

Ниже приведен пример одного из вариантов программы при выполнении лабораторной работы.

```
; Пример программы для изучения внешних прерываний
.include "m8535def.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 USART
      reti
      .org $00C  ; вектор Буфер Передатчика пуст USART
      reti
      .org $00D  ; вектор передатчика USART
      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

      ldi r16,$f0  ; выводы PD0, PD1, PD2 и PD3 конфигурировать как
      out DDRD,r16  ; входы, а остальные выводы порта D как выходы

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

      ldi r16,$ff  ; выводы порта B конфигурировать как выходы
      out DDRB,r16

m1:   ; основной цикл
      sbis PINA,2  ; опросить PA2. От состояния этого входа
                   ; зависит разрешение глобального прерывания
      rjmp m2
      sei          ; разрешение глобального прерывания
      ldi r16,$c0
```

```
     out GICR,r16  ; разрешение прерываний INT0 и INT1
     rjmp m3
m2:  cli ; запрет глобального прерывания

m3:  rcall wait
     in r16,PINA  ; опросить тумблеры порта A
     lsr r16  ; сдвинуть разряды регистра r16 четыре раза вправо
     lsr r16
     lsr r16
     lsr r16
     ldi r17,$0f
     and r16,r17  ; биты ISC01, ISC00, ISC11, ISC10 принимают
     out  MCUCR,r16  ; соответственно состояние PA7, PA6, PA5, PA4

          ; !!! команды программы по индивидуальному заданию
     ldi r16,0b00001000 ; пример - зажечь светодиод VD4,
     out portB,r16      ; остальные погасить
     rjmp m1; переход на начало основного цикла

pr0: ; ***Подпрограмма обработки прерывания INT0***
     in r16,SREG
     push r16      ; сохранение SREG
          ; !!! команды, реализующие индивидуальное задание
     ldi r16,0b00000100 ; например зажечь светодиод VD3,
     out portB,r16      ; остальные погасить
     ; сброс флага прерывания по INT0 - бит 6 рег.GIFR
     in r24,GIFR
     ldi r25,0b10111111
     and r24,r25
     out GIFR,r24
     pop r16
     out SREG,r16
     reti

pr1: ; ***Подпрограмма обработки прерывания INT1***
     in r16,SREG
     push r16      ; сохранение SREG
          ; !!! команды, реализующие индивидуальное задание
     ldi r16,0b00010000 ; например зажечь светодиод VD5
     out portB,r16      ; остальные погасить
     ; сброс флага прерывания по INT1 - бит 7 рег.GIFR
     in r24,GIFR
     ldi r25,0b01111111
     and r24,r25
     out GIFR,r24
     pop r16
     out SREG,r16
     reti

wait:      ; подпрограмма задержки
     push r20;
     push r21;
     ldi r20,$0ff
```

```
loop1: ldi r21,$0ff
loop: dec r21
     brne loop
     dec r20
     brne loop1
     pop r21;
     pop r20;
     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. Включение тумблера PA2 разрешает глобальное прерывание, тумблеры PA7 и PA6 обеспечивают задание характера сигнала прерывания INT0 (по фронтам или по уровню сигнала), тумблеры PA5 и PA4 аналогично обеспечивают задание характера сигнала прерывания INT1.

Подпрограмма wait реализует задержку времени (примерно 25 мс). В некоторых вариантах требуются существенно большие времена задержки, а также изменение времени задержки по прерыванию. Для этого можно увеличить число используемых в подпрограмме регистров (как в подпрограмме wait, приведенной в лабораторной работе № 1), а также изменять по прерыванию числа, заносимые в регистры.

Варианты индивидуальных заданий

1. При включении микроконтроллера загорается светодиод VD1. По прерыванию INT0 светодиод VD1 гаснет, загорается светодиод VD2 (идет сдвиг влево). То есть по прерыванию INT0 последовательно загораются и гаснут светодиоды VD1, VD2, VD3, VD1, VD2 и т.д. По прерыванию INT1 сдвиг осуществляется вправо.

2. Фоновая программа реализует «бегущий» огонь на светодиодах VD1, VD2, VD3 слева направо, то есть поочередно загораются светодиоды VD1, VD2, VD3, VD1, VD2 и т.д. По прерыванию INT0 скорость «бега» в два раза увеличивается, а по прерыванию INT1 – в два раза уменьшается.

3. Фоновая программа реализует «бегущий» огонь на сегментах индикатора HG2, то есть 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. В исходном состоянии на индикаторе HG2 загорается цифра 0. По прерыванию INT0 увеличивается высвечиваемая цифра 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. Фоновая программа реализует работу светофора, то есть чередование сигналов следующее: зеленый (VD1), желтый (VD2), красный (VD3), желтый, зеленый и т.д. По прерыванию INT0 время горения каждого сигнала в два раза увеличивается, а по прерыванию INT1 – в два раза уменьшается.

6. В исходном состоянии на индикаторе HG2 горит цифра 8. По прерыванию INT0 цифра на индикаторе увеличивается на две единицы, по прерыванию INT1 – уменьшается на две единицы. При достижении цифры E прерывание INT0 не меняет эту цифру, при достижении цифры 0 прерывание INT1 также не меняет эту цифру.

7. При включении микроконтроллера фоновая программа обеспечивает вывод в порт B комбинации сигналов, соответствующей индикации на индикаторе HG2 цифры 5. По прерыванию INT0 содержимое порта B инвертируется. По прерыванию INT1 содержимое порта B инкрементируется. В отчете отразить получившиеся изображения на индикаторе.

8. При включении микроконтроллера в порт D выдается число 20h, в порт C – число 66h. По прерыванию INT0 – производить циклический сдвиг содержимого порта D вправо, по прерыванию INT1 – влево.

9. На индикаторе HG2 последовательно изменяются цифры от 0 до 9: 0, 1, 2, …, 9, 0, 1, … и т.д. По прерыванию INT0 скорость изменения цифр в два раза увеличивается, а по прерыванию INT1 – в два раза уменьшается.

10. В исходном состоянии на индикаторе HG1 горит цифра 1. По прерыванию INT0 индикатор HG1 гаснет, а на индикаторе HG2 загорается цифра 2. По следующему прерыванию INT0 на HG3 загорается цифра 3, затем на HG4 цифра 4. После этого прерывание INT0 перестает переключать индикаторы и цифры. По прерыванию INT1 последовательность переключения индикаторов и изменения цифр меняется на противоположную (до зажигания на индикаторе HG1 цифры 0).

11. В исходном состоянии в порт B выводится нулевой код (все светодиоды VD1 – VD8 погашены). По прерыванию INT0 этот код на единицу увеличивается, по прерыванию INT1 – на единицу уменьшается.

12. При включении микроконтроллера четыре светодиода (VD1 – VD4) горят, а другие четыре (VD5 – VD8) погашены. По прерыванию INT0 происходит циклический сдвиг горящих светодиодов на одну позицию влево, а по прерыванию INT1 – на одну позицию вправо.

13. Фоновая программа обеспечивает последовательное зажигание на индикаторах HG1 – HG4 цифры 8 (когда один из индикаторов горит, остальные погашены). По прерыванию INT0 скорость зажигания индикаторов в два раза увеличивается, а по прерыванию INT1 – в два раза уменьшается.

14. В исходном состоянии на индикаторах HG1 и HG2 горят цифры 0. По прерыванию INT0 цифра на HG1 на единицу увеличивается, по прерыванию INT1 цифра на HG2 на единицу увеличивается. После цифры 9 следует снова цифра 0.

15. Фоновая программа обеспечивает подачу непрерывного звукового сигнала постоянной частоты на звукогенератор HA1. По прерыванию INT0 частота звукового сигнала в два раза увеличивается, а по прерыванию INT1 – в два раза уменьшается.

16. Фоновая программа обеспечивает периодические щелчки звукогенератора HA1. По прерыванию INT0 частота щелчков в два раза увеличивается, а по прерыванию INT1 – в два раза уменьшается.

6.4. Лабораторная работа № 3. Изучение программирования таймеров/счетчиков

Цель работы

Изучение функционирования таймеров/счетчиков микроконтроллера ATmega8535, получение практических навыков в их программировании.

Пояснения к работе

Микроконтроллер ATmega8535 семейства AVR имеет три таймера/счетчика. Два из них таймер/счетчик 0 и таймер/счетчик 2 8-разрядные, а таймер/счетчик 1 16-разрядный.

Таймеры/счетчики 0 и 1 используют для формирования своих тактовых сигналов выходы ступени деления общего 10- разрядного предварительного делителя тактовой частоты микроконтроллера. Они могут использоваться как таймеры с встроенной временной базой или как счетчики, переключаемые по состоянию на внешнем выводе соответственно PB0 (T0) и PB1 (T1).

Таймер/счетчик 2 имеет свой предварительный 10-разрядный делитель и может асинхронно тактироваться сигналом с вывода PC6 (T0SC1), что позволяет использовать таймер/счетчик 2 в качестве часов реального времени. Тогда генератор с частотой 32768 кГц подсоединяется между выводами PC6 (T0SC1) и PC7 (T0SC2).

Таймеры/счетчики выполнены в виде суммирующих счетчиков (в режиме Phase correct PWM они работают как реверсивные счетчики) и в любой момент имеют возможность чтения/записи. Если в таймер/счетчик занесено некоторое значение и выбран источник тактового сигнала, то таймер/счетчик продолжает счет с записанного значения с соответствующей тактовой частотой. При переполнении таймера/счетчика формируется запрос прерывания (если оно разрешено).

Таймеры/счетчики 0 и 2 имеют по одному, а таймер/счетчик 1 – два (A и B) регистра сравнения. Когда содержимое счетчика становится равным содержимому регистра сравнения, генерируется соответствующее прерывание (если оно разрешено). Кроме того, при наступлении этого события может изменяться состояние соответствующего вывода микроконтроллера: PB3 (OC0), BD7 (OC2), PD5 (OC1A) или PD4 (OC1B). Чтобы таймеры/счетчики могли управлять состоянием этих выводов, они должны быть сконфигурированы как выходы.

Таймер/счетчик 1 имеет в своем составе регистр захвата, назначение которого – сохранение в определенный момент времени состояния таймера/счетчика. Это действие может производиться либо по активному фронту сигнала на выводе ICP микроконтроллера, либо по сигналу от аналогового компаратора. Одновременно с записью в регистр захвата генерируется запрос на прерывание (если оно разрешено). Для захвата по сигналу с вывода ICP этот вывод должен быть сконфигурирован как вход. Если же он будет сконфигурирован как выход, то захват можно осуществлять программно, управляя соответствующим битом порта.

Таймеры/счетчики имеют различные режимы работы, в том числе позволяют реализовать режимы широтно-импульсных модуляторов (ШИМ).

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

Микроконтроллер ATmega8535 имеет регистр масок прерываний по таймерам/счетчикам TIMSK и регистр флагов прерываний по таймерам/счетчикам TIFR.

Регистр масок прерываний по таймерам/счетчикам TIMSK

Бит	7	6	5	4	3	2	1	0	
S39 ($59)	OCIE2	TOIE2	TICIEl	OCIElA	OCIE1B	TOIE1	OCIE0	TOIE0	**TIMSK**
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	

- Бит 7 – **OCIE2: Timer/Counter2 Output Compare Interrupt Enable** – **Разрешение прерывания по совпадению таймера/счетчика 2.** При установленном бите OCIE2 и установленном бите I регистра статуса разрешается прерывание по совпадению содержимого регистра выходного сравнения OCR2 (Output Compare Register 2) и состояния таймера/счетчика 2. Вектор прерывания – $003.
- Бит 6 – **TOIE2: Timer/Counter2 Overflow Interrupt Enable** – **Разрешение прерывания по переполнению таймера/счетчика 2.** При установленном бите T01E2 и установленном бите I регистра статуса разрешается прерывание по переполнению таймера/счетчика 2. Вектор прерывания – $004.
- Бит 5 – **TICIEl: Timer/Counterl Input Capture Interrupt Enable** – **Разрешение прерывания по захвату таймера/счетчика.** При установленном бите TICIEl и установленном бите I регистра статуса разрешается прерывание по захвату таймера/счетчика 1 сигналом по выводу PD6 (ICP). Вектор прерывания – $005.
- Бит 4 – **OCIE1A: Timer/Counterl Output CompareA Match Interrupt Enable** – **Разрешение прерывания по совпадению регистра A с таймером/счетчиком 1.** При установленном бите OCIE1A и установленном бите I регистра статуса разрешается прерывание по совпадению регистра выходного сравнения OCR1A с состоянием таймера/счетчика 1. Вектор прерывания – $006.
- Бит 3 – **OCIE1B: Timer/Counterl Output CompareB Match Interrupt Enable** – **Разрешение прерывания по совпадению регистра B с таймером/счетчиком 1.** При установленном бите OCIE1B и установленном бите I регистра статуса разрешается прерывание по совпадению регистра выходного сравнения OCR1B с состоянием таймера/счетчика 1. Вектор прерывания – $007.
- Бит 2 – **TOIE1: Timer/Counter1 Overflow Interrupt Enable** – **Разрешение прерывания по переполнению таймера/счетчика 1.** При установленном бите TOIE1 и установленном бите I регистра статуса разрешается прерывание по переполнению таймера/счетчика 1. Вектор прерывания – $008.
- Бит 1 – **OCIE0: Timer/Counter0 Output Compare Match Interrupt Enable** – **Флаг разрешения прерывания по совпадению таймера/счетчика 0.**

При установленном бите OCIE0 и установленном бите I регистра статуса разрешается прерывание по совпадению регистра выходного сравнения OCR0 с состоянием таймера/счетчика 0. Вектор прерывания – $013.

- Бит 0 – **TOIEO: Timer/Counter0 Overflow Interrupt Enable – Разрешение прерывания по переполнению таймера/счетчика 0.** При установленном бите TOIEO и установленном бите I регистра статуса разрешается прерывание по переполнению таймера/счетчика 0. Вектор прерывания – $009.

При возникновении какого-либо прерывания в регистре флагов TIFR устанавливается соответствующий флаг.

Регистр флагов прерываний по таймерам/счетчикам TIFR

Бит	7	6	5	4	3	2	1	0	
$38 ($58)	OCF2	TOV2	ICFl	OCFlA	OCF1B	TOV1	OCF0	TOV0	**TIFR**
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	

- Бит 7 – **OCF2: Output Compare Flag 2 – Флаг совпадения таймера/счетчика 2 и данных OCR2.** Бит OCF2 устанавливается при совпадении состояния таймера/счетчика 2 и содержимого регистра OCR2.
- Бит 6 – **TOV2: Timer/Counter2 Overflow Flag – Флаг переполнения таймера/счетчика 2.** Бит TOV2 устанавливается при переполнении таймера/счетчика 2. В режиме PWM (широгно-импульсного модулятора) этот бит устанавливается при смене таймером/счетчиком 2 направления счета при переходе через $00.
- Биты 5 – **ICF1: Input Capture Flag 1 – Флаг входного захвата таймера/счетчика 1.** Бит ICF1 устанавливается в случае захвата входа и показывает, что состояние таймера/счетчика 1 переслано во входной регистр захвата ICR1.
- Бит 4 – **OCF1A: Output Compare Flag 1A – Флаг совпадения выхода 1A.** Бит OCF1A устанавливается при совпадении состояния таймера/счетчика 1 и содержимого регистра OCR1A.
- Бит 3 – **OCF1B: Output Compare Flag 1B – Флаг совпадения выхода 1B.** Бит OCF1B устанавливается при совпадении состояния таймера/счетчика 1 и содержимого регистра OCR1B.
- Бит 2 – **TOV1: Timer/Counterl Overflow Flag – Флаг переполнения таймера/счетчика 1.** Бит TOV1 устанавливается при переполнении таймера/счетчика 1. В режиме PWM этот бит устанавливается при смене таймером/счетчиком 1 направления счета при переходе через $00.
- Бит 1 – **OCF0: Output Compare Flag 0 – Флаг прерывания по событию «Совпадение» таймера/счетчика 0.** Бит OCF0 устанавливается при совпадении состояния таймера/счетчика 0 и содержимого регистра OCR0.
- Бит 0 – **TOV0: Timer/Counter0 Overflow Flag – Флаг переполнения таймера/счетчика 0.** Бит TOV0 устанавливается при переполнении таймера/счетчика 0.

Каждое из прерываний по таймерам/счетчикам выполняется при установленном бите I в регистре SREG, установленных бите разрешения прерывания и

соответствующем флаге прерывания. Флаг прерывания аппаратно сбрасывается при обработке соответствующего вектора прерывания. Возможна очистка бита записью во флаг логической единицы. Программно установить флаг таймера/счетчика, записав в него логическую единицу, невозможно.

8-разрядный таймер/счетчик 0 осуществляет счет в регистре TCNT0.

Таймер/счетчик 0 - TCNT0

Бит	7	6	5	4	3	2	1	0	
$32 ($52)	MSB							LSB	**TCNT0**
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	

Аналогичный формат имеет регистр сравнения таймера/счетчика 0 OCR0, его адрес – $3C ($5C).

Таймер/счетчик 0 управляется регистром управления TCCR0.

Регистр управления таймером/счетчиком 0 – TCCR0

Бит	7	6	5	4	3	2	1	0	
S33 ($53)	FOC0	WGM00	COM0l	COM00	WGM01	CS02	CS01	CS00	**TCCR0**
Чтение/Запись	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	

- Бит 7 – **FOC0: Force Output Compare – Принудительное изменение состояния вывода OC0 (режимы Normal и CTC).** При записи лог. 1 в этот разряд состояние вывода OC0 изменяется в соответствии установкам разрядов COM01:COM00. Прерывание при этом не генерируется и сброс таймера (в режиме CTC) не производится. В режимах Fast PWM и Phase Correct PWM этот разряд должен быть сброшен в «0». При чтении разряда всегда возвращается «0».

- Биты 6, 3 – **WGM00:01: Waveform Generation Mode – Режим работы таймера/счетчика.** Эти разряды определяют режим работы таймера/счетчика 0 в соответствии с табл. 14:

Таблица 14. Режимы работы таймера/счетчика 0

Номер режима	WGM01	WGM00	Название режима
0	0	0	Normal
1	0	1	Phase correct PWM
2	1	0	CTC (сброс при совпадении)
3	1	1	Fast PWM

- Биты 5, 4 – **COM01:00: Compare Output Mode – Режим работы блока сравнения.** Эти разряды определяют поведение вывода OC0 при наступлении события «Совпадение». Влияние содержимого этих разрядов на состояние вывода зависит от режима работы таймера/счетчика.

- Биты 2, 1, 0 – **CS02, CS01, CS00: Clock Select – Выбор тактовой частоты.** Эти биты позволяют выбрать источник тактового сигнала для таймера/счетчика 0: системный тактовый сигнал ТС с частотой f_{TC}, масштабированный тактовый сигнал с выхода определенной ступени предварительного делителя частоты или внешний сигнал, поступающий на вход T0, а также запускать

и останавливать таймер/счетчик 0 (см. табл. 15). Если таймер/счетчик 0 используется как счетчик, то вывод T0 конфигурируется как вход.

Таблица 15. Выбор источника тактового сигнала

CS02	CS01	CS00	Описание
0	0	0	Таймер/счетчик 0 остановлен
0	0	1	f_{TC}
0	1	0	f_{TC} / 8
0	1	1	f_{TC} / 64
1	0	0	f_{TC} / 256
1	0	1	f_{TC} / 1024
1	1	0	Внешний вывод T0, падающий фронт
]	1	1	Внешний вывод T0, нарастающий фронт

Наиболее простой режим работы таймера/счетчика 0 – режим Normal. В этом режиме счетный регистр TCNT0 функционирует как обычный суммирующий счетчик, по каждому импульсу тактового сигнала осуществляется его инкрементирование. При переходе через значение $FF возникает переполнение, и счет продолжается со значения $00. Флаг прерывания по переполнению TOV0 при этом устанавливается в 1. При равенстве счетного регистра и регистра сравнения OCR0 устанавливается соответствующий флаг прерывания OCF0 и, если бит OCIE0 регистра маски установлен в 1, генерируется прерывание. Наряду с установкой флага при равенстве счетного регистра и регистра сравнения может изменяться состояние вывода OC0 микроконтроллера. Каким образом оно будет изменяться, определяется битами COM01:COM00 регистра управления TCCR0 в соответствии с табл. 16.

Таблица 16. Управление выводом OC0 в режимах Normal и CTC

COM01	COM00	Описание
0	0	Таймер/счетчик 0 отключен от вывода OC0
0	1	Состояние вывода меняется на противоположное
1	0	Вывод сбрасывается в 0
1	1	Вывод устанавливается в 1

В режиме CTC счетный регистр тоже функционирует как обычный суммирующий счетчик, инкрементирование которого осуществляется по каждому импульсу тактового сигнала. Однако максимально возможное значение счетного регистра и, следовательно, разрешающая способность счетчика определяются регистром сравнения OCR0. После достижения значения, записанного в регистре сравнения, счет продолжается со значения $00. При достижении счетчиком максимального значения устанавливается флаг OCF0 и, если бит OCIE0 регистра маски установлен в 1, генерируется прерывание. Одновременно с установкой флага может изменяться состояние вывода OC0 микроконтроллера в зависимости от значений бит COM01:COM00 в соответствии с табл. 16. Путем использования переключения вывода OC0 в этом режиме в противоположное состояние можно формировать сигнал заданной частоты (в зависимости от содержимого регистра сравнения).

Режим Fast PWM («Быстродействующий ШИМ») позволяет генерировать высокочастотный сигнал с широтно-импульсной модуляцией. В связи с высокой частотой генерируемого сигнала данный режим с успехом может использоваться в таких приложениях, как регулирование мощности, выпрямление, цифро-аналоговое преобразование и др. Счетный регистр в этом режиме функционирует как суммирующий счетчик, инкрементирование которого осуществляется по каждому импульсу тактового сигнала. Состояние счетчика изменяется от $00 до максимального значения $FF, после чего счетный регистр сбрасывается и цикл повторяется. При достижении счетчиком максимального значения устанавливается флаг прерывания по переполнению TOV0 в регистре флагов, а при равенстве содержимого счетного регистра и регистра сравнения OCR0 устанавливается флаг OCF0. Особенностью работы схемы сравнения в этом режиме является двойная буферизация записи в регистр OCR0, которая заключается в том, что записываемое число на самом деле сохраняется в специальном буферном регистре, а изменение содержимого регистра сравнения происходит только в момент достижения счетчиком максимального значения (рис. 9). Благодаря такому решению исключается появление несимметричных импульсов сигнала (помех) на выходе модулятора, которые были бы неизбежны при непосредственной записи в регистр сравнения. Состояние вывода OC0 микроконтроллера в этом режиме также определяется содержимым битов COM01:COM00 регистра TCCR0 (см. табл. 17).

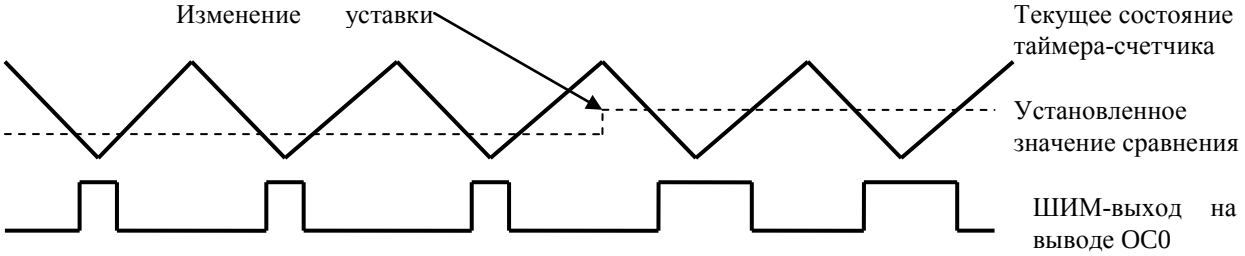


Рис. 9. Синхронизированная фиксация уставки в OCR0

Таблица 17. Управление выводом OC0 в режимах PWM

COM01	COM00	Описание
0	0	Таймер/счетчик 0 отключен от вывода OC0
0	1	Таймер/счетчик 0 отключен от вывода OC0
1	0	Сбрасывается в 0 при равенстве регистров TCNT0 и OCR0, устанавливается в 1 при сбросе счетчика – в режиме Fast PWM или по совпадению при обратном счете – в режиме Phase Correct PWM (неинвертированный ШИМ)
1	1	Устанавливается в 1 при равенстве регистров TCNT0 и OCR0, сбрасывается в 0 при сбросе счетчика – в режиме Fast PWM или по совпадению при обратном счете – в режиме Phase Correct PWM (инвертированный ШИМ)

Режим Phase Correct PWM («ШИМ с точной фазой»), как и режим Fast PWM, предназначен для генерации сигналов с широтно-импульсной модуляцией. Однако в этом режиме счетный регистр функционирует как реверсивный

счетчик, изменение состояния которого осуществляется по каждому импульсу тактового сигнала. Состояние счетчика сначала изменяется от $00 до максимального значения $FF, а затем обратно до $00. Соответственно максимальная частота сигнала в этом режиме в 2 раза меньше максимальной частоты сигнала в режиме Fast PWM. Тем не менее благодаря «симметричности» изменения состояния счетчика режим Phase Correct PWM предпочтительнее использовать для решения задач управления двигателями. При достижении счетчиком минимального значения ($00) также происходит смена направления счета и одновременно устанавливается флаг прерывания TOV0. При равенстве содержимого счетного регистра и регистра сравнения OCR0 устанавливается флаг OCF0 и изменяется состояние вывода OC0. Характер изменения определяется, как обычно, содержимым битов COM01:COM00 регистра TCCR0 (см. табл. 17). Для исключения несимметричных выбросов в этом режиме тоже реализована двойная буферизация записи в регистры сравнения. Поэтому действительное изменение содержимого регистра сравнения происходит только в момент достижения счетчиком максимального значения.

Таймер/счетчик 1 осуществляет счет в 16-разрядном регистре TCNT1, физически состоящем из двух регистров – старшего байта TCNT1H и младшего байта TCNT1L.

Таймер/счетчик 1 – TCNT1 (TCNT1H и TCNT1L)

Бит	15	14	13	12	11	10	9	8	
$2D ($4D)	MSB								**TCNT1H**
$2C ($4C)								LSB	**TCNT1L**
	7	6	5	4	3	2	1	0	
Чтение/Запись	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	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	
	0	0	0	0	0	0	0	0	

Этот регистр содержит текущее значение 16-разрядного таймера/счетчика 1. С тем чтобы CPU могло считывать/записывать и старший и младший байты этого регистра одновременно, обращение к нему реализовано посредством 8-разрядного регистра временного хранения TEMP.

- **Запись в таймер/счетчик 1 – TCNT1:**

Когда CPU производит запись в старший байт (TCNT1H) записываемые данные размещаются в регистре TEMP. Затем, когда CPU производит запись в младший байт (TCNT1L), данные младшего байта объединяются с байтом данных регистра TEMP и все 16 битов одновременно переписываются в регистр таймера/счегчика TCNT1. Следовательно, при 16-разрядных операциях записи обращение к старшему байту (TCNT1H) должно выполняться первым.

- **Чтение таймера/счетчика 1 – TCNT1:**

Когда CPU считывает младший байт (TCNT1L), то его содержимое направляется непосредственно, а содержимое старшего байта (TCNT1H) размещается в регистре TEMP. При считывании старшего байта его содержимое CPU принимает из регистра TEMP. Следовательно, при 16-разрядных операциях чтения первым должно выполняться обращение к младшему байту (TCNT1L).

Аналогично счетному регистру устроены два 16-разрядных регистра сравнения таймера/счетчика 1. Регистр сравнения OCR1A состоит из старшего байта OCR1AH с адресом $2B ($4B) и младшего байта OCR1AL с адресом $2A ($4A). Регистр сравнения OCR1B состоит из старшего байта OCR1BH с адресом $2B ($4B) и младшего байта OCR1BL с адресом $2A ($4A). Отличие 16-разрядного регистра захвата ICR1, состоящего из старшего байта ICR1H с адресом $27 ($47) и младшего байта ICR1L с адресом $26 ($46), состоит в том, что он обеспечивает только чтение его содержимого. При обращении к 16-разрядным регистрам сравнения OCR1A, OCR1B и захвата ICR1 также, как и при обращении к TCNT1, используется регистр временного хранения TEMP. Если основная программа и подпрограммы обработки прерываний используют обращение к указанным регистрам, то прерывания на время обращения должны быть запрещены.

16-разрядный таймер/счетчик 1 управляется двумя регистрами TCCR1A и TCCR1B.

Регистр управления A таймера/счетчика 1 – TCCR1A

Бит	7	6	5	4	3	2	1	0	
$2F ($4F)	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	**TCCR1A**
Чтение/Запись	R/W	R/W	R/W	R'W	W	W	R/W	R/W	
Исх. значение	0	0	0	0	0	0	0	0	

- Биты 7, 6 – **COM1A1, COM1A0: Compare Output Mode A – Режим работы блока сравнения A.** Эти биты определяют поведение вывода OC1A при совпадении таймера/счетчика 1 с регистром OCR1A. В зависимости от режима работы таймера/счетчика 1 состояние вывода изменяется аналогично таймеру/счетчику 0 в соответствии с табл. 16 и 17.

Биты 5, 4 – **COM1B1, COM1B0: Compare Output Mode B – Режим работы блока сравнения B.** Эти биты определяют поведение вывода OC1B при совпадении таймера/счетчика 1 с регистром OCR1B. В зависимости от режима работы таймера/счетчика 1 состояние вывода изменяется аналогично таймеру/счетчику 0 в соответствии с табл. 16 и 17.

- Биты 3, 2 – **FOC1A, FOC1B: Force Output Compare – Принудительное изменение состояния выводов OC1A и OC1B.** При записи в эти разряды логической единицы состояние выводов OC1A и OC1B изменяется в соответствии с установками разрядов COM1A1:COM1A0 и COM1B1:COM1B0. Прерывание при этом не генерируется и сброс таймера (в режиме CTC) не производится. Эта функция доступна только в режимах, которые не используются для генерации сигнала с ШИМ. При чтении разрядов всегда возвращается «0».
- Биты 1, 0 – **WGM11, WGM10: Waveform Generation Mode – Режим работы таймера/счетчика.** Данные биты совместно с битами WGM13 и WGM12 регистра TCCR1B определяют режим работы таймера/счетчика, как это показано в табл. 18.

Регистр управления B таймера/счетчика 1 – TCCR1B

Бит	7	6	5	4	3	2	1	0	
$2E($4E)	ICNC1	ICES1	—	WGM13	WGM12	CS12	CS11	CS10	**TCCR1B**
Чтение/Запись	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Исходное значение	0	0	0	0	0	0	0	0	

- Бит 7 – **ICNC1: Input Capturel Noise Canceler (4 CKs) – Установка режима подавления шума на входе захвата.** При сброшенном в состояние 0 бите ICNC1 функция подавления шума входного триггера захвата запрещена. Вход захвата переключается по первому нарастающему/падающему фронту, поступившему на вывод входа захвата ICP. При установленном в состояние 1 бите ICNC1 выполняются четыре последовательных опроса состояния вывода ICP и все четыре выборки должны иметь одинаковый (высокий/низкий), определяемый битом ICES1, уровень. Частота опроса соответствует частоте тактового сигнала.
- Бит 6 – **ICES1: Input Capturel Edge Select – Выбор фронта срабатывания на входе захвата.** При сброшенном в состояние 0 бите ICES1 содержимое таймера/счетчика 1 по падающему фронту на выводе входа захвата ICP пересылается в регистр входного захвата ICR1. При установленном в 1 бите ICES1 содержимое таймера/счетчика 1 пересылается в регистр входного захвата ICR1 по нарастающему фронту на выводе входа захвата ICP.
- Бит 5 – **Res: Reserved Bit – Зарезервированный бит.** Данный бит в микроконтроллерах ATmega8535 зарезервирован и при считывании всегда покажет состояние 0.
- Биты 4, 3 – **WGM13, WGM12: Waveform Generation Mode – Режим работы таймера/счетчика**. Совместно с разрядами WGM11, WGM10 регистра TCCR1A определяют режим работы таймера/счетчика 1, как это показано в табл. 18.
- Биты 2, 1, 0 – **CS12, CS11, CS10: Clock Select – Выбор тактовой частоты.** Эти биты позволяют выбрать источник тактового сигнала для таймера/счетчика 1: системный тактовый сигнал TC, масштабированный тактовый сигнал с выхода определенной ступени предварительного делителя частоты или внешний сигнал, поступающий на вход T1, а также запускать и останавливать таймер/счетчик 1, аналогично соответствующим настройкам для таймера/счетчика 0 (см. табл. 15). Если таймер/счетчик 1 используется как счетчик, то вывод T1 конфигурируется как вход.

В режиме Normal таймер/счетчик 1 функционирует как обычный суммирующий счетчик. При переходе через значение $FFFF возникает переполнение, устанавливается флаг прерывания по переполнению TOV1, и счет продолжается со значения $0000. Оба блока сравнения таймера в этом режиме могут использоваться как для генерации прерываний, так и для формирования сигналов. Состояние выходов OC1A и OC1B каждого из блоков сравнения определяется содержимым битов COM1A1, COM1A0 и COM1B1, COM1B0 регистр TCCR1A аналогично таймеру/счетчику 0 в соответствии с табл. 16.

В режиме CTC (сброс при совпадении) счетный регистр тоже функционирует как обычный суммирующий счетчик. Однако максимально возможное значение счетного регистра и, следовательно, разрешающая способность счетчика определяются либо регистром сравнения OCR1A (режим 4), либо регистром захвата ICR1 (режим 12). После достижения максимального значения счет продолжается со значения $0000. Флаг прерывания TOV1 устанавливается

при изменении значения счетного регистра с $FFFF на $0000. При достижении счетчиком максимального значения устанавливается флаг OCF1A, если модуль счета определяется регистром сравнения OCR1A, или ICFl, если модуль счета определяется регистром захвата ICR1. Одновременно с установкой соответствующих флагов может изменяться состояние выводов OC1A и OC1B микроконтроллера (см. табл. 16).

Таблица 18. Режимы работы таймера/счетчика 1

Номер режима	WGM13	WGM12	WGM11	WGM10	Название режима	Модуль счета (TOP)	Обновление регистров OCR1A и OCR1B	Момент установки флага TOV1
0	0	0	0	0	Normal	$FFFF	Немедленно	$FFFF
1	0	0	0	1	Phase correct PWM, 8-битный	$00FF	При TOP	$0000
2	0	0	1	0	Phase correct PWM, 9-битный	$01FF	При TOP	$0000
3	0	0	1	1	Phase correct PWM, 10-битный	$03FF	При TOP	$0000
4	0	1	0	0	CTC (сброс при совпадении)	OCR1A	Немедленно	$FFFF
5	0	1	0	1	Fast PWM, 8-битный	$00FF	При TOP	При TOP
6	0	1	1	0	Fast PWM, 9-битный	$01FF	При TOP	При TOP
7	0	1	1	1	Fast PWM, 10-битный	$03FF	При TOP	При TOP
8	1	0	0	0	Phase and Frequency Correct PWM	ICR1	$0000	$0000
9	1	0	0	1	Phase and Frequency Correct PWM	OCR1A	$0000	$0000
10	1	0	1	0	Phase correct PWM	ICR1	При TOP	$0000
11	1	0	1	1	Phase correct PWM	OCR1A	При TOP	$0000
12	1	1	0	0	CTC (сброс при совпадении)	ICR1	Немедленно	$FFFF
13	1	1	0	1	Зарезервировано	-	-	-
14	1	1	1	0	Fast PWM	ICR1	При TOP	При TOP
15	1	1	1	1	Fast PWM	OCR1A	При TOP	При TOP

Режим Fast PWM («Быстродействующий ШИМ») позволяет генерировать высокочастотный сигнал с широтно-импульсной модуляцией. Отличие от одноименного режима 8-битных таймеров/счетчиков заключается в том, что 16-битный таймер/счетчик позволяет генерировать ШИМ-сигнал различной разрядности. Состояние счетчика инкрементируется от $0000 до максимального значения, после чего счетный регистр сбрасывается и цикл повторяется. В зависимости от номера режима максимальное значение счетчика (разрешение ШИМ-сигнала) либо является фиксированным значением (режимы 5, 6 и 7), либо определяется содержимым регистров таймера/счетчика ICR1 (режим 14)

или OCR1A (режим 15). Разрешающая способность в режимах 14 и 15 переменная и может изменяться от 2 до 16 разрядов (модуль счета может иметь значение от $0003 до $FFFF). При работе с какими-либо фиксированными значениями модуля счета для задания модуля рекомендуется использовать регистр захвата, при этом регистр OCR1A может использоваться для формирования ШИМ-сигнала. Если же частота ШИМ-сигнала может часто меняться, то для задания модуля счета рекомендуется использовать регистр сравнения. В этом случае за счет буферизации записи в регистры сравнения исключается появление несимметричных импульсов сигнала на выходе.

При достижении счетчиком максимального значения устанавливается флаг прерывания TOV1. Одновременно с ним в режиме 14 устанавливается флаг ICF1, а в режиме 15 – флаг OCF1A. При равенстве содержимого счетного регистра и какого-либо регистра сравнения устанавливается соответствующий флаг прерывания и изменяется состояние соответствующего выхода (OC1A или OC1B). Состояние этих выходов определяется содержимым соответствующих битов COM согласно табл.17. Дополнительно к этой таблице в режимах 14 и 15 при значении битов COM1A1 = 0, COM1A0 = 1 состояние вывода OC1A при равенстве регистров TCNT1 и OCR1A меняется на противоположное.

В режиме Phase Correct PWM счетный регистр функционирует как реверсивный счетчик, состояние которого сначала изменяется от $0000 до максимального значения, а затем обратно до $0000. Соответственно, максимальная частота ШИМ-сигнала в этом режиме в 2 раза ниже максимальной частоты сигнала в режиме Fast PWM. В зависимости от номера режима максимальное значение счетчика (разрешение ШИМ-сигнала) либо является фиксированным значением (режимы 1, 2 и 3), либо определяется содержимым регистров таймера/счетчика ICR1 (режим 10) или OCR1A (режим 11). Разрешающая способность в режимах 14 и 15 переменная и может изменяться от 2 до 16 разрядов (модуль счета может иметь значение от $0003 до $FFFF). При достижении счетчиком максимального значения происходит смена направления счета и производится обновление содержимого регистра сравнения. Если модуль счета определяется регистром сравнения ICR1 (режим 10) или OCR1A (режим 11), одновременно с обновлением регистра сравнения устанавливается флаг ICF1 либо OCF1A соответственно.

При достижении счетчиком минимального значения ($0000) также происходит смена направления счета и одновременно устанавливается флаг прерывания TOV1. При равенстве содержимого счетного регистра и какого-либо регистра сравнения устанавливается соответствующий флаг прерывания и изменяется состояние соответствующего выхода. Состояние этих выходов определяется содержимым соответствующих битов COM согласно табл. 17. Дополнительно к этой таблице в режимах 10 и 11 при значении битов COM1A1 = 0, COM1A0 = 1 состояние вывода OC1A при равенстве регистров TCNT1 и OCR1A меняется на противоположное.

Режим Phase and Frequency Correct PWM («ШИМ с точной фазой и частотой») очень похож на режим Phase Correct PWM. Единственная принципиальная разница — обновление содержимого регистра сравнения происходит в мо-

мент достижения счетчиком минимального значения. Максимальное значение счетчика (разрешение ШИМ-сигнала) в этом режиме может определяться только регистрами таймера/счетчика ICRl (режим 8) или OCR1A (режим 9), разрешающая способность переменная. Время прямого счета несмотря на изменение регистра сравнения всегда равно времени обратного счета, выходные импульсы симметричны, и соответственно частота генерируемого сигнала остается постоянной.

8-разрядный таймер/счетчик 2 осуществляет счет в регистре TCNT2, который имеет адрес $25 ($45). Таймер/счетчик 2 содержит 8-разрядный регистр сравнения OCR2 с адресом $23 ($43). Форматы этих регистров аналогичны форматам регистроф TCNT0 и OCR0 таймера/счетчика 0.

Таймер/счетчик 2 управляется регистром TCCR2.

Регистр управления таймером/счетчиком 2 – TCCR2

Бит	7	6	5	4	3	2	1	0	
$25($45)	FOC2	WGM20	COM21	CQM20	WGM21	CS22	CS21	CS20	**TCCR2**
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	

Назначение отдельных битов регистра TCCR2 аналогично назначению соответствующих битов регистра TCCR0 таймера/счетчика 0, кроме битов выбора тактовой частоты CS22, CS21 и CS20. Тактовый сигнал таймера/счетчика 2, имеющего асинхронный режим работы, может формироваться либо из системного тактового сигнала, либо — в асинхронном режиме — из сигнала от дополнительного кварцевого резонатора. При этом для подачи на счетный регистр может использоваться либо сам тактовый сигнал с частотой f_{TC}, либо масштабированный тактовый сигнал с выхода определенной ступени предварительного делителя частоты в соответствии с табл. 19. Также с помощью битов CS можно запускать и останавливать таймер/счетчик 2.

Таблица 19. Выбор источника тактового сигнала таймера/счетчика 2

CS02	CS01	CS00	Описание
0	0	0	Таймер/счетчик 2 остановлен
0	0	1	f_{TC}
0	1	0	f_{TC} / 8
0	1	1	f_{TC} / 32
1	0	0	f_{TC} / 64
1	0	1	f_{TC} / 128
1	1	0	f_{TC} / 256
]	1	1	f_{TC} / 1024

В режимах Normal, CTC, Fast PWM и Phase Correct PWM с тактированием системным тактовым сигналом таймер/счетчик 2 работает аналогично таймеру/счетчику 0. В асинхронном режиме непосредственная запись в регистры TCNT2, OCR2 и TCCR2 синхронизируется с тактовым сигналом таймера/счетчика. При записи числа в любой из указанных регистров оно сохраняется в специальном временном регистре, своем для каждого регистра таймера/счетчика. А пересылка содержимого временного регистра в рабочий регистр

таймера/счетчика осуществляется по третьему после записи положительному фронту сигнала на выводе TOSC1. Соответственно запись нового значения можно производить только после пересылки содержимого временного регистра в регистр таймера/счетчика. Иначе возможно повреждение прежнего содержимого регистра и генерация прерывания. Для управления асинхронным режимом предназначен регистр состояния асинхронного режима ASSR.

Регистр состояния асинхронного режима таймера/счетчика 2 – ASSR

Бит	7	6	5	4	3	2	1	0	
$22($42)	-	-	-	-	AS2	TCN2UB	OCR2UB	TCR2UB	**ASSR**
Чтение/Запись	R	R	R	R	R/W	R/W	R/W	R/W	
Исходное значение	0	0	0	0	0	0	0	0	

- Биты 7 – 4 – **Res: Reserved Bits – Зарезервированные биты.**
- Бит 3 – **AS2: Переключение режима работы.** Если бит установлен в 1, то на вход предделителя таймера/счетчика 2 поступают импульсы с кварцевого генератора таймера/счетчика (асинхронный режим). В этом режиме выводы TOSC1 и TOSC2 используются для подключения кварцевого резонатора и соответственно не могут использоваться как контакты ввода/вывода общего назначения. Если бит сброшен в 0, то на вход предделителя поступает внутренний тактовый сигнал микроконтроллера. В этом случае выводы TOSC1 и TOSC2 являются линиями ввода/вывода общего назначения.
- Бит 2 – **TCN2UB: Состояние обновления регистра TCNT2.** При записи в регистр TCNT2 этот флаг устанавливается в 1, а после пересылки записываемого значения из регистра временного хранения в данный регистр флаг аппаратно сбрасывается в 0. Таким образом, сброшенный флаг TCN2UB означает, что регистр TCNT2 готов для записи в него нового значения.
- Бит 1 – **OCR2UB: Состояние обновления регистра OCR2.** При записи в регистр сравнения этот устанавливается в 1, а после пересылки записываемого значения в регистр флаг аппаратно сбрасывается в 0. Сброшенный флаг OCR2UB означает, что регистр сравнения готов для записи в него нового значения.
- Бит 0 – **TCR2UB: Состояние обновления регистра TCCR2**. При записи в регистр управления этот флаг устанавливается в 1, а после пересылки записываемого значения в регистр флаг аппаратно сбрасывается в 0. Сброшенный флаг TCR2UB означает, что соответствующий регистр управления готов для записи в него нового значения.

В лабораторном комплексе изучение асинхронного режима работы таймер/счетчика 2 не предусмотрено.

При составлении программ необходимо учитывать, что частота кварцевого резонатора микроконтроллера – 8 МГц.

Варианты индивидуальных заданий

1. Запрограммировать таймер/счетчик 0 для 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.

2. Запрограммировать таймер/счетчик 0 для динамической индикации цифр на семисегментных индикаторах HG1 и HG2. Таймер/счетчик 1 запрограммировать для счета нажатий кнопки. На индикацию выводить в виде шестнадцатеричного числа содержимое регистра TCNT1L. При нажатии другой кнопки содержимое счетчика сбрасывается в нуль.

3. Запрограммировать «бегущий» огонь на индикаторах VD1 – VD3, используя таймер/счетчик 0. Переключение тумблера изменяет темп "бега". Для возможности визуального наблюдения "бега" огня необходимо реализовать программным путем счетчик числа формирований флага переполнения таймера/счетчика.

4. Запрограммировать таймер/счетчик 0 для работы в ШИМ режиме для выдачи импульсов на вывод PB3 (OC0) микроконтроллера, к которому подключен светодиод VD4. При включении одного тумблера скважность импульсов ШИМ уменьшается, а при включении другого – увеличивается. Уменьшение и увеличение скважности импульсов оценивается по свечению светодиода VD4.

5. Запрограммировать таймер/счетчик 1 для выдачи звукового сигнала на вывод PA1 микроконтроллера, к которому подключен звукоизлучатель HA1. При нажатии на кнопку частота сигнала уменьшается, при нажатии на другую кнопку – увеличивается. Уменьшение и увеличение частоты сигнала оценивается по изменению тона звучания звукоизлучателя.

6. Запрограммировать микроконтроллер для реализации на основе таймеров звукового сигнала линейно меняющейся частоты (10 ступеней) на звукоизлучателе HA1. После достижения максимальной частоты происходит ступенчатый переход на минимальную частоту.

7. На основе таймеров организовать с дискретностью 1 секунда измерение длительности включенного состояния тумблера (INT0). Максимально допустимая длительность 9 секунд. Индикацию секунд организовать на семисегментном индикаторе HG2. При нажатии кнопки происходит обнуление индикатора.

8. Таймер/счетчик 0 и таймер/счетчик 1 запрограммировать на счет внешних событий. На индикаторе HG2 индицировать разность содержимого счетчиков. Если число в таймере/счетчике 0 меньше, чем в таймере/счетчике 1, то на индикаторе высвечивается цифра 0. Если число в таймере/счетчике 0 превышает больше чем на 9 число в таймере/счетчике 1, то высвечивается цифра 9. В остальных случаях высвечивается цифра превышения числа в таймере/счетчике 0 над числом в таймере/счетчике 1. При нажатии кнопки счетчики сбрасываются (обнуляются).

9. Запрограммировать таймер/счетчик 1 для работы в ШИМ-режиме для выдачи импульсов на выводы PD4 (OC1B) и PD5 (OC1A) микроконтроллера, к которым подключены семисегментные индикаторы HG1 и HG2. На индикаторы выводятся цифры 0 и 9. Скважность импульсов ШИМ на индикаторе HG1 постепенно (в течение 15–20 секунд) уменьшается, а на индикаторе HG2 – увеличивается.

10. Запрограммировать таймер/счетчик 2 для динамической индикации кода, вводимого с помощью тумблеров PA2 – PA7, на семисегментных индикаторах HG3 и HG4. Код должен отображаться в виде шестнадцатеричного числа.

11. Организовать при помощи таймеров последовательное высвечивание цифр от 1 до 4 на индикаторах HG1 – HG4 с периодом, задаваемым тумблерами PA2 – PA3 (1, 2, 3 или 4 секунды).

12. Запрограммировать таймер/счетчик 0 для работы в ШИМ-режиме для выдачи импульсов на вывод PB3 (OC0) микроконтроллера, к которому подключен светодиод VD4. Скважность импульсов должна определяться кодом, вводимым с помощью тумблеров PA2 – PA4 (восемь вариантов скважности). Уменьшение и увеличение скважности импульсов оценивается по свечению светодиода VD4.

13. Запрограммировать микроконтроллер для реализации на основе таймеров коротких звуковых щелчков на звукоизлучателе HA1 с частотой, определяемой кодом, вводимым с помощью тумблеров PA5 – PA7 (восемь вариантов частоты).

14. Запрограммировать таймер/счетчик 1 для работы в ШИМ режиме для выдачи импульсов на выводы PD4 (OC1B) и PD5 (OC1A) микроконтроллера, к которым подключены семисегментные индикаторы HG1 и HG2. Скважность импульсов ШИМ на индикаторе HG1 задается с помощью тумблеров PA2 – PA3, а на индикаторе HG2 – с помощью тумблеров PA4 – PA5 (по 4 варианта скважности). На каждый из индикаторов выводятся цифры 0, 1, 2 или 3 в зависимости от варианта скважности подаваемых импульсов.

15. Запрограммировать таймер/счетчик 2 для динамической индикации двоичных цифр (0 или 1), вводимых с помощью тумблеров PA4 – PA7, на семисегментных индикаторах HG1 – HG4.

16. Запрограммировать таймер/счетчик 1 для выдачи звукового сигнала на вывод PA1 микроконтроллера, к которому подключен звукоизлучатель HA1. Частота сигнала, определяющая тон звучания звукоизлучателя, задается с помощью тумблеров PA2 – PA4 (восемь вариантов частоты).

6.5. Лабораторная работа № 4. Изучение аналого-цифрового преобразователя

Цель работы

Изучение функционирования встроенного аналого-цифрового преобразователя (АЦП) микроконтроллера ATmega8535, получение практических навыков программирования микроконтроллера для обработки аналоговых сигналов.

Пояснения к работе

Микроконтроллер ATmega8535 имеет АЦП (ADC – Analog to Digital Converter) со следующими характеристиками:

– разрешение 10 разрядов;

– точность ±2 LSB;

– интегральная нелинейность 0,5 LSB;

– время преобразования 70...280 мкс;

– 8 мультиплексируемых каналов входа;

– режимы циклического и однократного преобразования;

– прерывание по завершению ADC преобразования;

– устройство подавления шумов Sleep-режима.

АЦП подсоединен к 8-канальному аналоговому мультиплексору, позволяющему использовать любой вывод порта А в качестве входа АЦП. АЦП содержит усилитель выборки/хранения, удерживающий напряжение входа АЦП во время преобразования на неизменном уровне (см. рис. 10).

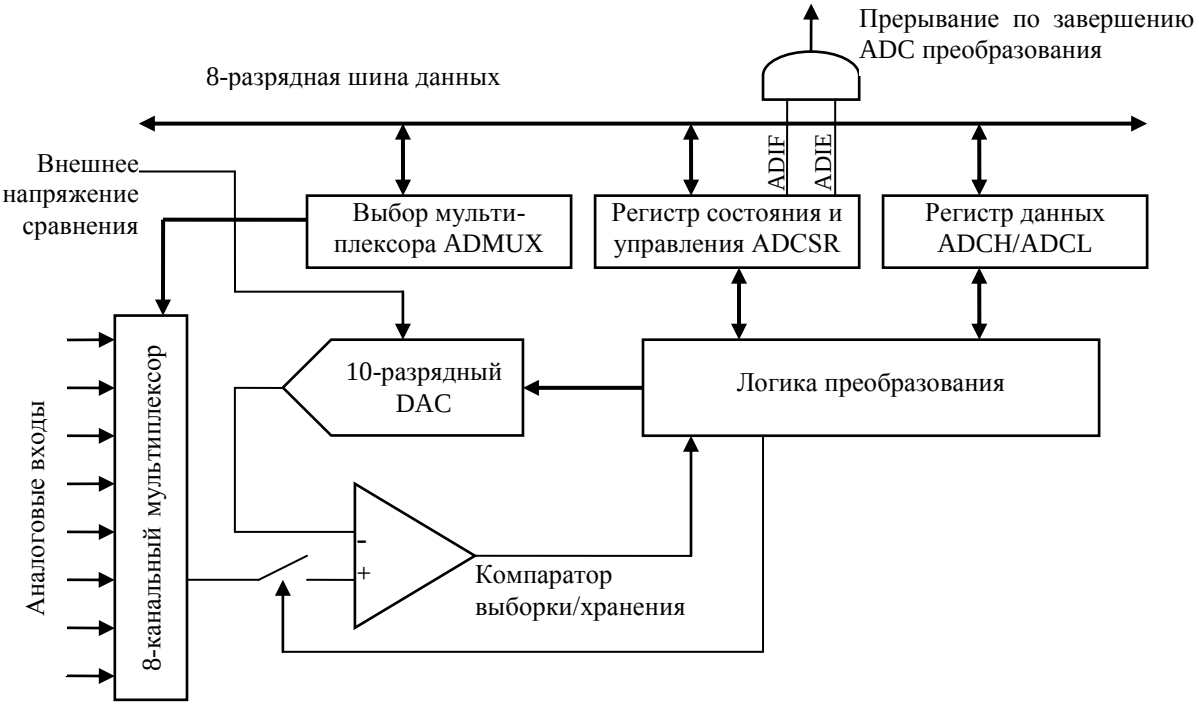


Рис. 10. Структурная схема аналого-цифрового преобразователя

Для питания АЦП используются два отдельных вывода: AVCC и AGND. Вывод AGND должен быть подсоединен к GND, а напряжение AVCC не должно отличаться от напряжения VCC более чем на ±0,3 В.

Внешнее напряжение сравнения подается на вывод AREF и должно быть в диапазоне от AGND до AVCC.

Аналого-цифровой преобразователь может работать в двух режимах: режиме однократного преобразования и режиме циклического преобразования. В режиме однократного преобразования каждое преобразование инициируется пользователем. В режиме циклического преобразования АЦП осуществляет выборку и обновление содержимого регистра данных АЦП непрерывно. Выбор режима производится битом ADATE регистра состояния и управления ADCSRA.

Работа АЦП разрешается установкой в состояние 1 бита ADEN в регистре ADCSRA. Первому преобразованию, начинающемуся после разрешения АЦП, предшествует пустое инициализирующее преобразование. На пользователе это отражается лишь тем, что первое преобразование будет занимать 26 тактовых циклов вместо обычных 14. Преобразование начинается с установки в состояние 1 бита начала преобразования ADSC. Этот бит находится в состоянии 1 в течение всего цикла преобразования и сбрасывается по завершению преобразования аппаратно. Если в процессе преобразования выполняется смена канала

данных, то АЦП вначале закончит текущее преобразование и лишь потом выполнит переход к другому каналу.

АЦП формирует 10-разрядный результат в двух регистрах – ADCH и ADCL. Для чтения правильного результата из этих регистров существует следующий механизм. При чтении результата первым должен читаться регистр ADCL. После этого доступ АЦП к регистрам данных блокируется. Это обозначает, что если следующее преобразование завершено между чтением ADCL и ADCH, его результат будет потерян. После чтения регистра ADCH доступ АЦП к регистрам данных восстанавливается.

Регистры данных ADC – ADCL и ADCH

Бит	15	14	13	12	11	10	9	8	
$05 ($25)	—	—	—	—	—	—	ADC9	ADC8	**ADCH**
$04 ($24)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	**ADCL**
	7	6	5	4	3	2	1	0	
Чтение/Запись	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Исходное значение	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

АЦП имеет свой флаг и вектор прерывания. Флаг запроса ADIF устанавливается при завершении преобразования.

АЦП включает делитель частоты, который формирует для него тактовый сигнал из синхросигнала процессора. АЦП работает с тактовой частотой в диапазоне от 50 до 250 кГц.

Биты ADPS0 – ADPS2 регистра состояния и управления ADCSR используются для формирования тактовой частоты АЦП из сигнала XTAL. Делитель частоты работает, когда установлен бит ADEN.

При запуске АЦП установкой бита ADSC преобразование начинается по заднему фронту импульса синхросигнала АЦП. Один такт синхросигнала требуется на выборку-сохранение аналогового сигнала, после чего 13 циклов затрачивается на собственно преобразование и запись результата в регистры ADCL, ADCH.

Регистр выбора мультиплексора ADC – ADMUX

Бит	7	6	5	4	3	2	1	0	
$07 ($27)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	**ADMUX**
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	

- Биты 7, 6 – **REFS1:REFS0: Reference Selection Bits – Выбор источника опорного напряжения.** Модуль АЦП может использовать различные источники опорного напряжения. Выбор конкретного источника опорного напряжения осуществляется с помощью разрядов REFS1:REFS0 регистра ADMUX (см. табл. 20). Как указано в таблице, внутренний источник опорного напряжения подключается к выводу AREF микроконтроллера. Поэтому при его использовании к выводу AREF можно подключить внешний фильтрующий конденсатор для повышения помехозащищенности.

Таблица 20. Выбор источника опорного напряжения

REFS 1	REFS 0	Источник опорного напряжения (ИОН)
0	0	Внешний ИОН, подключенный к выводу AREF, внутренний ИОН отключен
0	1	Напряжение питания AVCC
1	0	Зарезервировано
1	1	Внутренний ИОН напряжением 2,56 В, подключенный к выходу AREF

• Бит 5 **– ADLAR: ADC Left Adjust Result – Выравнивание результата преобразования влево.** По умолчанию результат преобразования выравнивается вправо (старшие 6 разрядов регистра ADCH незначащие). Однако он может выравниваться и влево (младшие 6 разрядов регистра ADCL незначащие). Для управления выравниванием результата преобразования служит разряд ADLAR регистра ADMUX. Если этот разряд установлен в «1», результат преобразования выравнивается по левой границе 16-разрядного слова, если обращен в «0» – по правой границе.

• Биты 4.. 0 **– MUX4.. MUX0: Analog Channel Select Bits 2-0 – Биты выбора аналогового канала.** Состояние данных битов определяет, какой из восьми аналоговых каналов (0 – 7) будет подключен к АЦП, а также различные варианты подключения каналов с дифференциальным входом, в том числе с предварительным усилением входного сигнала с коэффициентом 10 или 200.

Регистр управления и состояния ADC - ADCSRA

Бит	7	6	5	4	3	2	1	0	
$06 ($26)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	**ADCSR**
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	

• Бит 7 – **ADEN: ADC Enable – Разрешение АЦП**. Установка данного бита в состояние 1 разрешает работу АЦП. Очистка бита запрещает работу АЦП. Запрещение АЦП в процессе преобразования прекращает преобразование.

• Бит 6 – **ADSC: ADC Start Conversion – Запуск преобразования**. В режиме однократного преобразования для запуска каждого цикла преобразования необходимо устанавливать бит ADSC в состояние 1. В циклическом режиме бит ADSC устанавливается в состояние 1 только при запуске первого цикла преобразования. Каждый раз после первой установки бита ADSC, выполненной после разрешения ADC или одновременно с разрешением ADC, будет выполняться пустое преобразование, предшествующее активируемому преобразованию. Это пустое преобразование активирует АЦП. ADSC будет сохранять состояние 1 в течение всего цикла преобразования и сбрасывается по завершении преобразования.

• Бит 5 – **ADATE: ADC Auto Trigger Enable – Выбор режима работы АЦП**. В микроконтроллере Atmega8535 запуск АЦП возможен не только по команде пользователя, но и по прерыванию от некоторых периферийных

устройств, имеющихся в составе микроконтроллера. Для выбора режима работы используется разряд ADATE регистра ADCSRA и разряды ADTS2..0 регистра специальных функций SFIOR.

Если разряд ADATE сброшен в «0», АЦП работает в режиме одиночного преобразования. Если же разряд ADATE установлен в «1», функционирование АЦП определяется содержимым разрядов ADTS2...0 согласно табл. 21.

Таблица 21. Источник сигнала для запуска преобразования Atmega8535

ADTS2	ADTS1	ADTS0	Источник стартового сигнала
0	0	0	Режим непрерывного преобразования
0	0	1	Прерывание от аналогового компаратора
0	1	0	Внешнее прерывание INT0
0	1	1	Прерывание по событию «Совпадение» таймера/счетчика 0
1	0	0	Прерывание по переполнению таймера/счетчика 0
1	0	1	Прерывание по событию «Совпадение B» таймера/счетчика 1
1	1	0	Прерывание по переполнению таймера/счетчика 1
1	1	1	Прерывание по событию «Захват» таймера/счетчика 1

• **Бит 4 - ADIF: ADC Interrupt Flag - Флаг прерывания АЦП.** Данный бит устанавливается в состояние 1 по завершению преобразования и обновления регистров данных. Прерывание по завершению преобразования АЦП выполняется, если в состояние 1 установлены бит ADIE и I-бит регистра SREG. Бит ADIF сбрасывается аппаратно при выполнении подпрограммы обработки соответствующего вектора прерывания. Кроме того, бит ADIF может быть очищен записью во флаг логической единицы.

• **Бит 3 – ADIE: ADC Interrupt Enable - Разрешение прерывания от АЦП.** При установленных в состояние 1 бите ADIE и I-бите регистра SREG активируется прерывание по завершению преобразования АЦП.

• **Биты 2…0 – ADPS2…ADPS0: ADC Prescaler Select Bits – Выбор коэффициента предварительного деления.** Данные биты определяют коэффициент деления частоты XTAL для получения необходимой тактовой частоты АЦП (см. табл. 22).

Таблица 22. Выбор коэффициента предварительного деления

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

В блоке управления лабораторного комплекса канал PA0 порта A используется для ввода сигнала на АЦП микроконтроллера. Вводимое напряжение снимается с потенциометрического резистора, движок которого выведен на лицевую панель. Максимальное напряжение, снимаемое с потенциометрического резистора, составляет 5 В.

Ниже представлена программа, обеспечивающая вывод на семисегментный индикатор HG1 цифр от 0 до 5 в зависимости от напряжения на входе АЦП.

Цифра 0 выводится, если напряжение на входе АЦП находится в пределах 0...0,5 В,

1 – в пределах 0,5...1,5 В,

2 – в пределах 1,5...2,5 В,

3 – в пределах 2,5...3,5 В,

4 – в пределах 3,5...4,5 В,

5 – в пределах 4,5...5,0 В.

```
; Пример использования АЦП микроконтроллером ATmega8535
.include "m8535def.inc" ; включить файл описания ATmega8535
.def code = r20 ; регистр куда помещается преобразованное
                ; число
.ORG$0  ; Обработка прерывания сброса
rjmp   Reset
.ORG$0e  ; Вектор АЦП
rjmp   inADC
;
Reset:
ldi  r16,$02  ; Инициализация вершины стека по адресу
out  SPH,r16  ; $025F
ldi  r16,$5f
out  SPL,r16
cbi  DDRA,0  ; PA0 на ввод
sbi  PORTA,0  ; Подключить вывод PA0
ldi  r16,$0ff   ; Определить все выводы портов D и C
out  DDRD,r16  ; на выход
out  DDRC,r16
sbi  PORTD,4  ; Активизировать семисегментный индикатор HG1
;
ldi  r16,$00  ; выбор нулевого канала АЦП (PA0)
out  ADMUX,r16
ldi  r16,$0e8 ;инициализация АЦП: разрешение и запуск циклич.
out  ADCSRA,r16 ;преобразования без деления тактовой частоты
sei  ; установка общего флага прерываний
m1:
rjmp m1  ; рабочий цикл программы, ничего не делаем
;
inADC:  ; Подпрограмма обслуживающая прерывания
        ; по запросу АЦП
in   r16,ADCL  ; Считать младший байт регистра данных АЦП
in   code,ADCH  ; Считать старший байт регистра данных АЦП
lsr  r16 ; Исключить два младших разряда
```

```
lsr  r16 ; преобразованного сигнала
swap code ; Поменять местами биты 3...0 с
lsl  code ; битами 7...4 и сдвинуть влево на
lsl  code ; два разряда
andi code,$0c0  ; Выделить биты 7 и 6 code
or  code,r16  ; Получить в code восьмиразрядное число
rcall kod  ; Вызвать подпрограмму зажигания цифры
reti
;
kod: ; Подпрограмма зажигания цифры
ldi  r22,51  ; В r22 дискрета напряжения 1 В на PA0 для цифр
             ; на семисегментном индикаторе
ldi  r21,26  ; В r21 максимальное значение кода АЦП при
             ; котором на индикаторе горит 0 - порог нуля
cp  code,r21 ; Если код АЦП не превышает порог нуля, то
brlo k0      ; зажечь 0
add  r21,r22 ; Установить порог единицы
cp  code,r21 ; Если код АЦП не превышает порог единицы, то
brlo k1      ; зажечь 1
add  r21,r22 ; Установить порог двойки
cp  code,r21 ; Если код АЦП не превышает порог двойки, то
brlo k2      ; зажечь 2
add  r21,r22 ; Установить порог тройки
cp  code,r21 ; Если код АЦП не превышает порог тройки, то
brlo k3      ; зажечь 3
add  r21,r22 ; Установить порог четверки
cp  code,r21 ; Если код АЦП не превышает порог четверки, то
brlo k4      ; зажечь 4
ldi  r23,$6d ; Если код АЦП выше порога четверки, то зажечь 5
out  PORTC,r23
ret  ; Выход из подпрограммы
;
k0:
ldi  r23,$3f  ; Зажечь 0
out  PORTC,r23
ret  ; Возврат
k1:
ldi  r23,$06  ; Зажечь 1
out  PORTC,r23
ret  ; Возврат
k2:
ldi  r23,$5b  ; Зажечь 2
out  PORTC,r23
ret  ; Возврат
k3:
ldi  r23, $4f  ; Зажечь 3
out  PORTC,r23
ret  ; Возврат
k4:
ldi  r23,$66  ; Зажечь 4
out  PORTC,r23
ret  ; Возврат
```

Варианты индивидуальных заданий

1. В одном из регистров общего назначения записано число, определяющее заданный уровень порога преобразуемого напряжения. Если напряжение на входе АЦП меньше (ниже) этого порога, то на семисегментном индикаторе HG2 высвечивается буква Н, при равенстве напряжений – буква Р, при превышении преобразуемым напряжением уровня порога (больше) – буква Б. При выполнении этого задания два младших разряда преобразованного числа АЦП опустить.

2. Полученный результат преобразования АЦП вывести в шестнадцатеричном формате на семисегментные индикаторы HG1 и HG2 (динамическая индикация). При изменении положения движка потенциометрического резистора наблюдать изменение информации на индикаторах HG1 и HG2. При выполнении задания два младших разряда преобразованного числа АЦП опустить.

3. В двух регистрах общего назначения записаны числа, определяющие заданные уровни порогов преобразуемого напряжения. В первом регистре число меньше, чем во втором регистре. Если напряжение на входе АЦП меньше (ниже) порога первого регистра, то на семисегментном индикаторе HG1 высвечивается буква Н. Если результат преобразования АЦП находится между двумя порогами, то он отображается на индикаторах HG1 и HG2 (динамическая индикация). Если же результат преобразования превышает порог второго регистра, то на индикатор HG1 выводится буква Б. При выполнении задания два младших разряда преобразованного числа АЦП опустить.

4. Организовать мигание светодиода VD6. В момент загорания светодиода раздается щелчок звукоизлучателя HA1. При увеличении уровня напряжения на входе АЦП, скорость мигания увеличивается, при уменьшении напряжения – скорость уменьшается.

5. В двух регистрах общего назначения записаны числа, определяющие заданные уровни порогов преобразуемого напряжения. В первом регистре число меньше, чем во втором регистре. Если напряжение на входе АЦП меньше порога первого регистра, то на семисегментных индикаторах HG1 и HG2 в шестнадцатеричном формате выводится число первого регистра. Если напряжение на входе АЦП больше порога второго регистра, то – число второго регистра. Если же напряжение на входе АЦП располагается между указанными порогами, то выводится результат преобразования АЦП.

6. Организовать бегущий огонь на семисегментном индикаторе HG2. При увеличении уровня напряжения на входе АЦП скорость бега на семисегментном индикаторе увеличивается, при уменьшении – скорость бега уменьшается.

7. В двух регистрах общего назначения записаны числа, определяющие заданные уровни порогов преобразуемого напряжения. В первом регистре число меньше, чем во втором регистре. Если напряжение на входе АЦП меньше (ниже) первого уровня, то горит 1 светодиод. Если оно находится между двумя уровнями, то горят 2 светодиода. Если же напряжение на входе АЦП выше второго уровня, то горят 3 светодиода.

8. Организовать бегущий огонь на светодиодах VD1 – VD8. При увеличении уровня напряжения на входе АЦП, скорость бега на светодиодах увеличивается. При уменьшении – скорость бега уменьшается.

9. Полученный результат преобразования АЦП вывести в виде двоичного кода на светодиоды VD1 – VD8. При изменении положения движка потенциометрического резистора наблюдать изменение кода на светодиодах VD1 – VD8. При выполнении задания два младших разряда преобразованного числа АЦП опустить.

10. Разбить весь диапазон напряжений на входе АЦП на 8 равных поддиапазонов. При попадании преобразуемого напряжения в какой-либо поддиапазон зажечь соответствующий светодиод (в порядке VD1, VD2, …, VD8).

11. Организовать формирование звука изменяющейся частоты с помощью звукоизлучателя HA1. При увеличении уровня напряжения на входе АЦП частота звука увеличивается, при уменьшении напряжения – уменьшается.

12. Восемь младших разрядов полученного результата преобразования АЦП вывести в шестнадцатеричном формате на семисегментные индикаторы HG1 и HG2 (динамическая индикация). При изменении положения движка потенциометрического резистора наблюдать изменение информации на индикаторах HG1 и HG2. При равенстве или превышении преобразованным числом АЦП значения 256 зажечь на индикаторе HG1 букву Б.

13. Разбить весь диапазон напряжений на входе АЦП на 16 равных поддиапазонов. При попадании преобразуемого напряжения в какой-либо поддиапазон высветить на семисегментном индикаторе HG4 соответствующую шестнадцатеричную цифру от 0 до F.

14. Разбить весь диапазон напряжений на входе АЦП на 4 равных поддиапазон. При попадании преобразуемого напряжения в какой-либо поддиапазон высвечивать на поочередно на семисегментных индикаторах HG1 – HG4 соответствующую цифру от 0 до 3. В каждом более высоком диапазоне скорость поочередного зажигания индикаторов увеличивается.

15. Разбить весь диапазон напряжений на входе АЦП на 8 равных поддиапазонов. При попадании преобразуемого напряжения в какой-либо поддиапазон зажечь соответствующее количество светодиодов – 1, 2, … , 8 (в порядке VD1, VD2, …, VD8).

16. Полученный результат преобразования АЦП вывести в десятичном формате на семисегментные индикаторы HG1 и HG2 (динамическая индикация). При изменении положения движка потенциометрического резистора наблюдать изменение информации на индикаторах HG1 и HG2. При выполнении задания четыре младших разряда преобразованного числа АЦП опустить.

6.6. Лабораторная работа № 5. Изучение систем автоматизации на базе микроконтроллеров

Цель работы

На базе микроконтроллера ATmega8535 реализовать систему автоматизации управления виртуальным объектом.

Пояснения к работе

Виртуальный объект реализуется на экране монитора ПЭВМ в виде мультипликации. Микроконтроллер подает управляющие команды, по которым механизмы виртуального объекта совершают заданные перемещения с заданной скоростью. Сигналы с датчиков виртуального объекта также передаются в микроконтроллер. Изображения механизмов виртуального объекта при их включении изменяют свой цвет, при срабатывании датчиков виртуального объекта их изображение также меняет свой цвет.

Для управления виртуальными объектами между ПЭВМ и микроконтроллером постоянно идет обмен информацией по последовательному интерфейсу USART. От ПЭВМ передается в микроконтроллер информация о состоянии датчиков виртуального объекта, а микроконтроллер выдает на объект управляющие команды.

В ПЭВМ хранятся 4 варианта виртуальных объектов:

– роботизированный комплекс;

– методическая печь (участок транспортировки изделий в методическую печь для их нагрева перед прокаткой);

– нагревательный колодец (механизмы управления крышкой нагревательного колодца обжимного прокатного стана);

– сортировка и пакетирование (участок сортировки и пакетирования годных и бракованных листов металла).

В качестве примера ниже рассмотрено управление механизмами напольно-крышечной машины нагревательного колодца обжимного прокатного стана. Упрощенный вид напольно-крышечной машины представлен на рис. 11.

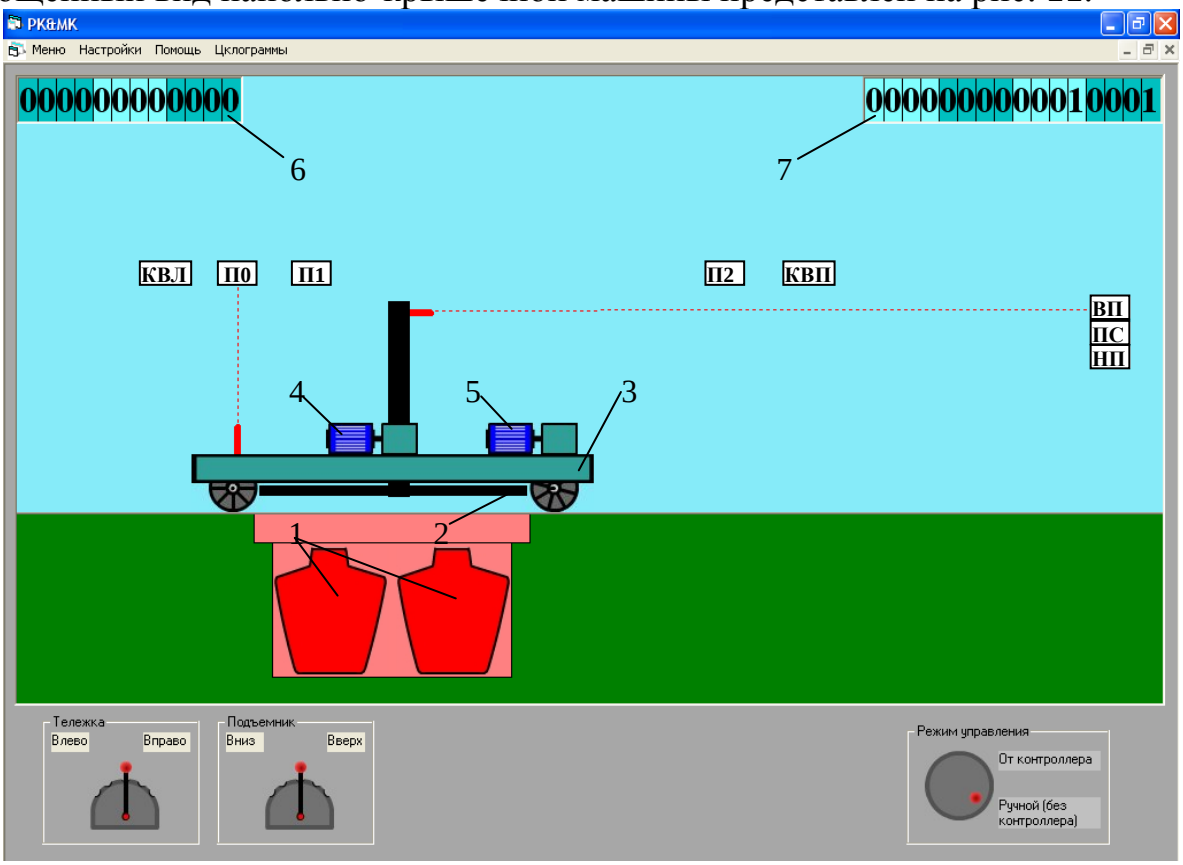


Рис. 11. Механизм управления крышкой нагревательного колодца

Это один из вариантов виртуальных объектов, представленных в приложении, где дано подробное описание назначения и последовательности работы напольно-крышечной машины. Здесь же повторяется конструктивная схема этого виртуального объекта и более кратко представлена последовательность работы механизмов объекта.

На рис. 11 кроме конструктивных механизмов объекта, расположения датчиков и исполнительных устройств (в рассматриваемом случае это электродвигатели подъема крышки 2 и передвижения тележки 3) представлены линейки битов управляющих команд на исполнительные устройства 6 и состояния датчиков объекта 7. Единица в бите линейки соответствует наличию команды и наличию сигнала с датчика.

Управление механизмами напольно-крышечной машины должно осуществляться с блока управления рабочего места студента. С этого блока должны подаваться для управления две ручные команды "Открыть" и "Закрыть". Принимаем решение, что команда "Открыть" подается на вход PA3 микроконтроллера, а "Закрыть" – на вход PA2.

В процессе управления объектом микроконтроллер должен считывать информацию из ПЭВМ о состоянии датчиков объекта, по программе формировать управляющие команды и выдавать их на объект для управления его механизмами. В ниже представленных программах управление указанным обменом осуществляется подпрограммой "irps". Для корректной передачи информации из ПЭВМ в микроконтроллер и обратно необходима также подпрограмма "delay".

Программирование виртуального объекта студентом заключается в считывании информации о состоянии датчиков из соответствующих регистров (in1, in2) и в выдаче в зависимости от этих данных управляющих сигналов в регистры (out1, out2). Биты регистров in1, in2, out1, out2 для рассматриваемого примера представлены в таблице 23, а для остальных виртуальных объектов – в приложении. Программа должна составляться с учетом этих битов. В программе студента кроме подпрограмм "irps" и "delay" должны быть правильно инициализированы порты, USART, настроены векторы прерываний. Преподаватель со своего рабочего места должен перед началом работы студента загрузить шаблон программы для виртуального объекта.

Таблица 23. Регистры микроконтроллера для обмена информацией с ПЭВМ

Мнемоника	Регистр	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
in 1	r20	КВП / kbp	П2 / n2	П1 / n1	П0 / n0	КВЛ / kbl	НП / np	ПС / ns	ВП / bp
in 2	r21	—	—	—	—	—	—	—	—
out 1	r23	—	—	Тележка медл. / tm	Влево / tl	Вправо / tp	Крышка медл. / km	Вниз / nn	Вверх / bb
out 2	r24	—	—	—	—	—	—	—	—

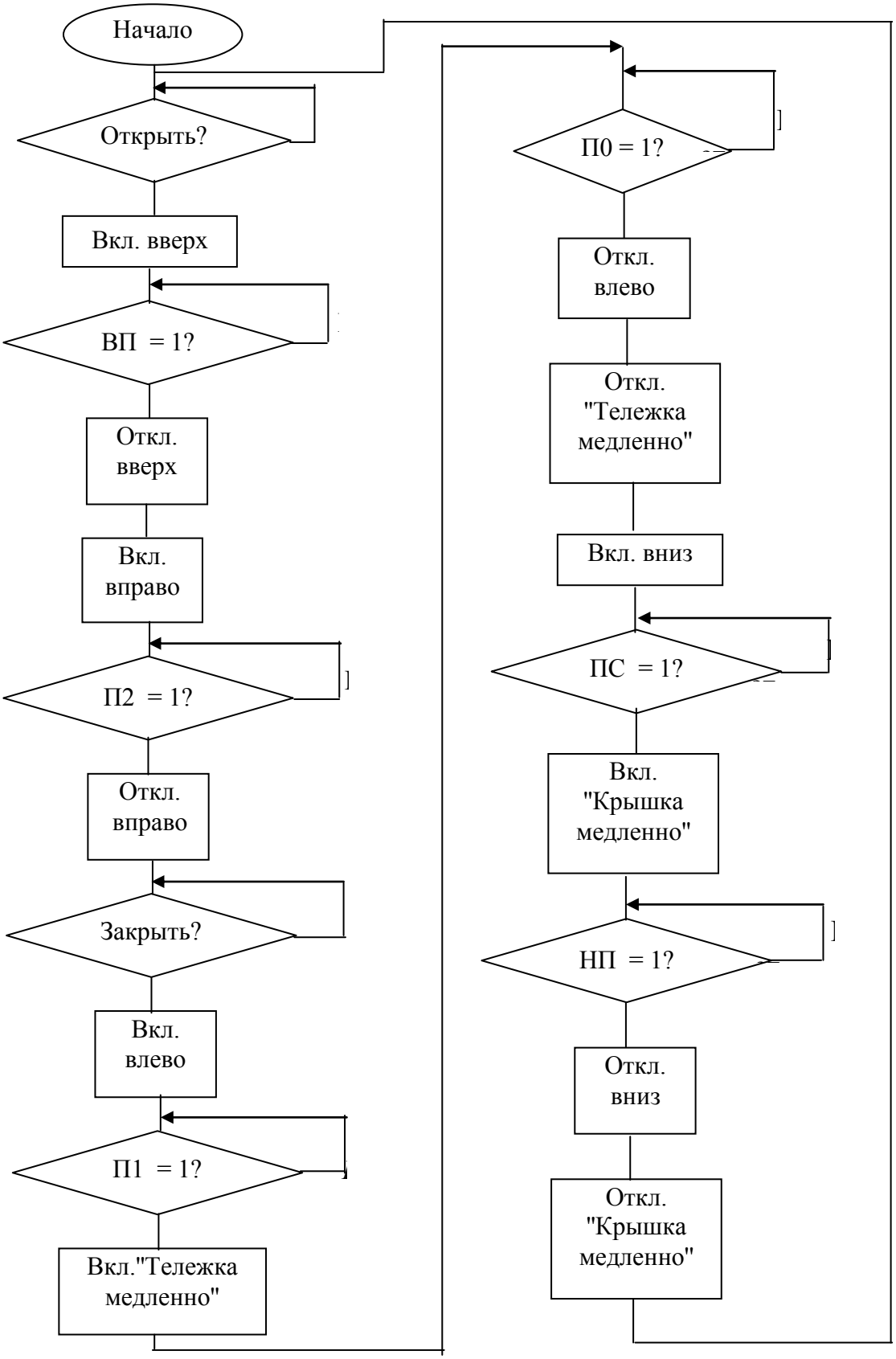


Рис.12. Схема алгоритма управления работой напольно-крышечной машины

На рис. 12 приведена схема алгоритма управления работой напольно-крышечной машины, соответствущая представленному в приложении описанию. Все действия однотипны: контролируется поступление необходимого сигнала, после чего выдаются требуемые управляющие команды.

Ниже представлена программа для микроконтроллера, реализующая приведенный алгоритм.

```
; Программа управления механизмами напольно-крышечной машины.
; Команды "Открыть" и "Закрыть" подаются кнопками PA3 и PA2
     .include"m8535def.inc"
     .def count=r18      ; Счетчик прочитанных слов
     .def temp=r19
     .def in1=r20   ; Регистры хранения состояния датчиков
     .def in2=r21   ; виртуального объекта автоматизации
     .def in3=r22   ; (читаемая информация)
     .def out1=r23  ; Регистры хранения управляющих команд на
     .def out2=r24  ; виртуальн. объект (передаваемая инф-ция)
;---------------- ------------------------
     .org $000 ; Вектор прерывания по Сбросу
     rjmp reset
     .org $00b ; Вектор прерывания по приему USART
     rjmp irps
     .org $00c ; Векст. прерыв. по пустому регистру данных USART
     reti
     .org $00d ; Вектор прерывания по завершению передачи USART
     reti
;---------------- ------------------------
reset:
     ldi R17,$ff    ; Установка указателя стека
     out spl,R17
     ldi R17,$00
     out sph,R17
;---------------- ------------------------
     sei  ; Установить флаг глобального прерывания в регистре SREG
     ldi  r16,$ff   ; Настроить порт C на выход
     out  ddrc,r16
     ldi  r16,00    ; Настроить порт A на вход
     out  ddra,r16
     ldi  r16,$ff   ; Настроить порт B на выход
     out  ddrb,r16
     ldi  r16,0xf0  ; Настроить порт D: биты 0...3 на вход,
     out  ddrd,r16  ; остальные на выход
     ldi R17,51     ; Инициализировать USART, скорость 9600 бод
     out ubrrl,R17
     ldi R17,0x9b   ; Разрешить прием - передачу 10011011
     out ucsrb,R17
;---------------- ------------------------
     ldi  count,0   ; Обнуление переменных
     ldi  temp,0
     ldi  in1,0
     ldi  in2,0
     ldi  in3,0
```

```
     ldi out1,0
     ldi out2,0
     ldi  r17,0
     out  portd,r17
;---------------- ------------------------
start:         ; Начало основной программы
     sei            ; Установить флаг глобального прерывания
     sbis pina,3    ; Проверка состояния кнопки "Открыть"
     rjmp start
     set            ; Установить флаг T в регистре SREG
     bld out1,0     ; Выдать команду "Вверх"
mo1: sei
     sbrs in1,0     ; Контроль состояния датчика ВП
     rjmp mo1
     clt            ; Очистить флаг T в регистре SREG
     bld out1,0     ; Отключить команду "Вверх"
     set            ; Установить флаг T в регистре SREG
     bld out1,3     ; Выдать команду "Вправо"
mo2: sei
     sbrs in1,6     ; Контроль состояния датчика П2
     rjmp mo2
     clt            ; Очистить флаг T в регистре SREG
     bld out1,3     ; Отключить команду "Вправо"
mo3: sei
     sbis pina,2    ; Проверка состояния кнопки "Закрыть"
     rjmp mo3
     set
     bld out1,4     ; Выдать команду "Влево"
mo4: sei
     sbrs in1,5     ; Контроль состояния датчика П1
     rjmp mo4
     set
     bld out1,5     ; Выдать команду "Тележка медленно"
mo5: sei
     sbrs in1,4     ; Контроль с остояния датчика П0
     rjmp mo5
     clt
     bld out1,4     ; Отключить команду "Влево"
     bld out1,5     ; Отключить команду "Тележка медленно"
     set
     bld out1,1     ; Выдать команду "Вниз"
mo6: sei
     sbrs in1,1     ; Контроль состояния датчика ПС
     rjmp mo6
     set
     bld out1,2     ; Выдать команду "Крышка медленно"
mo7: sei
     sbrs in1,2     ; Контроль состояния датчика НП
     rjmp mo7
     clt
     bld out1,1     ; Отключить команду "Вниз"
     bld out1,2     ; Отключить команду "Крышка медленно"
     rjmp start
```

```
;---------------- ----------------------
     irps:              ; Подпрограмма прерывания по приёму USART
     in  r25, sreg      ; Сохранение регистра  SREG
     cli  ; Запретить прерывания
     mov  count,temp
     sbi  portb,4   ; Включить светодиод VD5
     tst  count     ; Проверка счетчика на нуль
     brne      m21
     in  in2,udr    ; Записать первый принятый байт информации
     inc  count     ; с объекта
     mov  temp,count
     rjmp exit
m21: mov  count,temp
     cpi  count,1
     brne m22
     in  in1,udr    ; Записать второй принятый байт информации  mov
     count,temp     ; с объекта
     inc  count
     mov  temp,count
     rjmp      exit
m22: mov  count,temp
     in  in3,udr    ; Записать третий принятый байт информации
     ldi  count,0   ; с объекта
     mov  temp,count
     rcall delay
     out  udr,out2  ; Выдать первый байт команд на объект
     sbi  portb,5   ; Включить светодиод VD6
mt1: sbis ucsra,udre ; Передатчик готов к получению нового символа?
     rjmp      mt1
     ldi  r17,64
     out  ucsra,r17
     sbi  portb,6   ; Включить светодиод VD7
     rcall delay
     out  udr,out1  ; Выдать второй байт команд на объект
mt2: sbis ucsra,udre ; Передатчик готов к получению нового символа?
     rjmp mt2
     ldi  r17,64
     out  ucsra,r17
exit: mov count,temp
     sei
     out  sreg,r25  ; Восстановление регистра SREG
     cbi portb,4    ; Выключение индикации
     cbi portb,5
     cbi portb,6
     reti      ; Выход из подпрограммы прерывания
delay:         ; Подпрограмма временной задержки
     ldi r27,0x0F
loop1:   ldi   r28,0xFF
loop:    dec   r28
     brne loop
     dec r27
     brne loop1
     ret       ; Выход из подпрограммы
```

Другой вариант написания программы для микроконтроллера – по логическим уравнениям. Например, команда "Вверх" на движение крышки напольно-крышечной машины возникает при нажатии на кнопку "Открыть" (команда "Откр."). Возникшая команда "Вверх" сохраняется до прихода крышки в верхнее положение ВП. Этому содержательному описанию соответствует логическое уравнение

$$Вверх = (Откр. + Вверх)*\overline{ВП}.$$

Команда на движение крышки "Вниз" возникает после того как крышка побывала в правом положении П2 и вернулась в положение П0 над колодцем. Возникшая команда сохраняется до прихода крышки в крайнее нижнее положение НП. Представленному описанию соответствует логическое уравнение

$$Вниз = (П0*ВП*pn + Вниз)*\overline{НП}.$$

В этом уравнении промежуточная переменная pn выполняет роль памяти о том, что крышка побывала в положении П2. Она возникает в положении П2 и сохраняется до прохода крышки в положение НП. Тогда логическое выражение для рассматриваемой переменной имеет вид

$$pn = (П2 + pn)*\overline{НП}.$$

Команда на снижение скорости крышки Мк (крышка медленно) возникает только при движении "Вниз". Она появляется при наличии сигнала ПС и сохраняется, пока существует команда "Вниз". Логическое уравнение для команды Мк имеет вид

$$Мк = (ПС + Мк)*Вниз.$$

Команда на перемещение тележки "Вправо" возникает в верхнем положении крышки и нахождении тележки в положении П0, если тележка до этого не побывала в положении П2, т.е. если еще нет сигнала pn. Возникшая команда сохраняется до прихода тележки в положение П2. Команда должна отключаться, если по какой-то причине тележка дойдет до крайнего правого положения КВП (аварийное отключение). Логическое уравнение для этой команды имеет вид

$$Вправо = (П0*\overline{pn} + Вправо)*ВП*\overline{П2}*\overline{КВП}.$$

Команда на движение тележки "Влево" возникает в положении тележки П2 при нажатии на кнопку "Закрыть" (команда "Закр."). Возникшая команда сохраняется до прихода тележки в положение П0. Команда должна отключаться, если по какой-то причине тележка дойдет до крайнего левого положения КВЛ (аварийное отключение). Представленному описанию соответствует логическое уравнение

$$Влево = (П2*Закр + Влево)*ВП*\overline{ПО}*\overline{КВЛ}.$$

Команда Мт на снижение скорости тележки возникает только при движении тележки влево. Она появляется в положении тележки П1 и сохраняется, пока существует команда "Влево". Логическое уравнение для команды Мт имеет вид

Мт = (П1 + Мт)*Влево.

Пример программы микроконтроллера с использованием логических уравнений представлен ниже.

```
        ; Программа управления механизмами напольно-крышечной машины
        ; по логическим уравнениям
        .include "m8535def.inc"
        .def  count=r18
        .def  temp=r19
        ; ————————————————————————————————
        .def  in1=r20          ; Регистры хранения состояния датчиков и
        …                      ; управляющих команд (аналогично предыдущей
        .def  out2=r24         ; программе)
        ; ————————————————————————————————
        .def  A=r27       ; Регистр, используемый как аккумулятор A
        .def  B=r28       ;
        .def  C=r29       ; Регистр, используемый для инвертирования бита
        .def  D=r30       ; Хранение промежуточной переменной pn
        ; ————————————————————————————————
        .org  $0000            ; Вектора прерываний (аналогично предыдущей
        …                      ; программе)
        .org  $00d
        reti
        ; ————————————————————————————————
reset:
        ldi   R13,$ff          ; Установка указателя стека
        out   SPL,R13
        ldi   R13,$00
        out   SPH,R13
        ; ————————————————————————————————
        sei                    ; Настройка портов, инициализация UART
        …                      ; (аналогично предыдущей программе)
        out   UCR,R17
        ; ————————————————————————————————
        ldi   count,0          ; Обнуление переменных (аналогично предыду-
                              й
        …                      ; программе)
        out   portd,r17
        ; ————————————————————————————————
        ldi   A,0              ; Обнуление дополнительно введённых реги-
                              ров
        ldi   B,0
        ldi   C,1
        ldi   D,0
        ; ————————————————————————————————
start:                         ; Начало основной программы
        ;Программирование команды "Вверх"
        ; Вверх = (Открыть + Вверх) · НЕ ВП
        ; out1,0 = (pina,1 + out1,0) · НЕ in1,0
        in    A,pina           ; A,0 ← pina,3
```

```
lsr   A
lsr   A
lsr   A
bst   out1,0          ; B,0 ← out1,0
bld   B,0
or    A,B             ; A ← A ∨ B
bst   in1,0           ; B,0 ← in1,0
bld   B,0
eor   B,C             ; B ← B ⊕ C
and   A,B             ; A ← A • B
bst   A,0             ; out1,0 ← A,0
bld   out1,0          ; Выдать команду «Вверх»
; Программирование команды "Вниз"
```

; $\text{Вниз} = (\text{П0} \cdot \text{ВП} \cdot p_n + \text{Вниз}) \cdot \overline{\text{НП}}$

; $\text{out1,1} = (\text{in1,4} \cdot \text{in1,0} \cdot \text{D,0} + \text{out1,1}) \cdot \overline{\text{in1,2}}$

```
bst   in1,4           ; A,0 ← in1,4
bld   A,0
bst   in1,0           ; B,0 ← in1,0
bld   B,0
and   A,B             ; A ← A • B
bst   D,0             ; B,0 ← D,0
bld   B,0
and   A,B             ; A ← A • B
bst   out1,1          ; B,0 ← out1,1
bld   B,0
or    A,B             ; A ← A ∨ B
bst   in1,2           ; B,0 ← in1,2
bld   B,0
eor   B,C             ; B ← B ⊕ C
and   A,B             ; A ← A • B
bst   A,0             ; out1,1 ← A,0
bld   out1,1          ; Выдать команду «Вниз»
; Программирование промежуточной переменной pn
```

; $p_n = (\text{П2} + p_n) \cdot \overline{\text{НП}}$

; $\text{D,0} = (\text{in1,6} + \text{D,0}) \cdot \overline{\text{in1,2}}$

```
bst   in1,6           ; A,0 ← in1,6
bld   A,0
bst   D,0             ; B,0 ← D,0
bld   B,0
or    A,B             ; A ← A ∨ B
bst   in1,2           ; B,0 ← in1,2
bld   B,0
eor   B,C             ; B ← B ⊕ C
and   A,B             ; A ← A • B
bst   A,0             ; D,0 ← A,0
bld   D,0
```

```
; Программирование команды "Медленно крышка"
```
; $Мк = (ПС + Мк) \cdot Вниз$
```
; out1,2 = (in1,1 + out1,2) · out1,1
bst   in1,1          ; A,0 ← in1,1
bld   A,0
bst   out1,2         ; B,0 ← out1,2
bld   B,0
or    A,B            ; A ← A ∨ B
bst   out1,1         ; B,0 ← out1,1
bld   B,0
and   A,B            ; A ← A • B
bst   A,0            ; out1,2 ← A,0
bld   out1,2         ; Выдать команду «Медленно крышка»
; Программирование команды "Вправо"
```
; $Вправо = (П0 \cdot \overline{p_n} + Вправо) \cdot ВП \cdot \overline{П2} \cdot \overline{КВП}$
```
; out1,3 = (in1,4 · D,0 + out1,3) · in1,0 · in1,6 · in1,7
bst   in1,4          ; A,0 ← in1,4
bld   A,0
bst   D,0            ; B,0 ← D,0
bld   B,0
eor   B,C            ; B ← B ⊕ C
and   A,B            ; A ← A • B
bst   out1,3         ; B,0 ← out1,3
bld   B,0
or    A,B            ; A ← A ∨ B
bst   in1,0          ; B,0 ← in1,0
bld   B,0
and   A,B            ; A ← A • B
bst   in1,6          ; B,0 ← in1,6
bld   B,0
eor   B,C            ; B ← B ⊕ C
and   A,B            ; A ← A • B
bst   in1,7          ; B,0 ← in1,7
bld   B,0
eor   B,C            ; B ← B ⊕ C
and   A,B            ; A ← A • B
bst   A,0            ; out1,3 ← A,0
bld   out1,3         ; Выдать команду «Вправо»
; Программирование команды "Влево"
```
; $Влево = (П2 \cdot Закрыть + Влево) \cdot ВП \cdot \overline{П0} \cdot \overline{КВЛ}$
```
; out1,4 = (in1,6 · pina,2 + out1,4) · in1,0 · in1,4 · in1,3
bst   in1,6          ; A,0 ← in1,6
bld   A,0
in    B,pina         ; B,0 ← pina,2
lsr   B
```

```
        lsr   B
        and   A,B               ; A ← A • B
        bst   out1,4            ; B,0 ← out1,4
        bld   B,0
        or    A,B               ; A ← A ∨ B
        bst   in1,0             ; B,0 ← in1,0
        bld   B,0
        and   A,B               ; A ← A • B
        bst   in1,4             ; B,0 ← in1,4
        bld   B,0
        eor   B,C               ; B ← B ⊕ C
        and   A,B               ; A ← A • B
        bst   in1,3             ; B,0 ← in1,3
        bld   B,0
        eor   B,C               ; B ← B ⊕ C
        and   A,B               ; A ← A • B
        bst   A,0               ; out1,4 ← A,0
        bld   out1,4            ; Выдать команду «Влево»
        ; Программирование команды "Медленно тележка"
        ; Мт = (П1 + Мт) · Влево
        ; out1,5 = (in1,5 + out1,5) · out1,4
        bst   in1,5             ; A,0 ← in1,5
        bld   A,0               ;
        bst   out1,5            ; B,0 ← out1,5
        bld   B,0               ;
        or    A,B               ; A ← A ∨ B
        bst   out1,4            ; B,0 ← out1,4
        bld   B,0               ;
        and   A,B               ; A ← A • B
        bst   A,0               ; out1,5 ← A,0
        bld   out1,5            ; Выдать команду «Медленно тележка»

        rjmp  start
        ; ————————————————————————————————
irps:                           ; Подпрограмма прерывания по приему UART
        in    r25,sreg
                                ; Команды, реализующие последовательную
        …                       ; передачу (аналогично предыдущей программе)
        cbi   portd,6
        reti
        ; ————————————————————————————————
delay:                          ; Подпрограмма временной задержки
        ldi   r27,0xFF
                                ; Команды, реализующие временную задержку
        …                       ; (аналогично предыдущей программе)
        ret
```

Для проверки работоспособности программы управления виртуальным объектом после компиляции и записи программы в контроллер студент должен перевести тумблер «Режим» в положение «Авт». После этого у преподавателя появляется окно выбора виртуального объекта. Преподаватель должен выбрать соответствующий варианту студента объект. Кнопками и тумблерами, в зависимости от задачи, даются необходимые команды, и визуально контролируется работа механизмов. Если в работе обнаружены отклонения от данной последовательности, студент корректирует свою программу и аналогичным образом проверяет ее работоспособность. Для возврата к редактированию преподаватель закрывает окно с виртуальным объектом, после чего необходимо перевести тумблер «Режим» в положение «Ред».

Варианты индивидуальных заданий

В ПЭВМ реализовано 4 варианта виртуальных объектов. Описание этих объектов представлено в приложении. Для каждого варианта задачи имеется по 3 подварианта, которые требуют использования ограниченного количества сигналов датчиков и управляющих команд.

1. Вариант 1. Роботизированный комплекс.
2. Роботизированный комплекс. Подвариант 1.1.
3. Роботизированный комплекс. Подвариант 1.2.
4. Роботизированный комплекс. Подвариант 1.3.
5. Вариант 2. Методическая печь.
6. Методическая печь. Подвариант 2.1.
7. Методическая печь. Подвариант 2.2.
8. Методическая печь. Подвариант 2.3.
9. Вариант 3. Нагревательный колодец.
10. Нагревательный колодец. Подвариант 3.1.
11. Нагревательный колодец. Подвариант 3.2.
12. Нагревательный колодец. Подвариант 3.3.
13. Вариант 3. Сортировка и пакетирование.
14. Сортировка и пакетирование. Подвариант 4.1.
15. Сортировка и пакетирование. Подвариант 4.2.
16. Сортировка и пакетирование. Подвариант 4.3.

6.7. Приложение. Варианты технологических объектов управления

Ниже представлены варианты технологических объектов управления (объектов автоматизации), записанные в ПЭВМ. Для каждого варианта представлена конструктивная схема объекта в том виде, в котором она вызывается на экран дисплея при выборе данного варианта. На схеме показаны все необходимые датчики и исполнительные элементы, достаточные для функционирования системы в соответствии с поставленной задачей, и обучающиеся должны пользоваться именно указанными на рисунках (на экране) обозначениями элементов.

На экране монитора над изображением механизма представлены разряды (биты) команд на исполнительные механизмы и разряды (биты) сигналов датчиков. Они позволяют контролировать изменение команд и сигналов в процессе работы механизмов. Наличие команды или сигнала сопровождается появлением в соответствующем разряде 1, отсутствие команды или сигнала – 0.

Аналогичную информацию получает пользователь при анализе рисунка объекта управления. В частности, если фон исполнительного элемента и датчика серый (синий, белый), то это соответствует отсутствию команды и сигнала. Если фон красный (желтый) – соответствует включенному состоянию.

В табл. 24 представлено соответствие буквенно-цифровых позиционных обозначений сигналов датчиков в описании вариантов задач с сигналами ПЭВМ, подаваемыми в микроконтроллер, и с их адресами в микроконтроллере. В табл. 25 представлена аналогичная информация для команд управления исполнительными механизмами.

Таблица 24. Сигналы датчиков

Номер бита (датчика) в ПЭВМ	Адрес бита в ОЗУ микроконтроллера	№ 1 Роботизированный комплекс	№ 2 Методическая печь	№ 3 Нагревательный колодец	№ 4 Сортировка и пакетирование
X0	in1,0	К1	ПР0	ВП	Г
X1	in1,1	К2	ПР1	ПС	Б
X2	in1,2	К3	ПР2	НП	КВ
X3	in1,3	К4	ПР3	КВЛ	ПС
X4	in1,4	К5	ПТ0	П0	ПМ
X5	in1,5	К6	ПТ1	П1	КЛ
X6	in1,6	К7	ПТ2	П2	П1
X7	in1,7	К8	ПТ3	КВП	П2
X8	in2,0		ПЗ0		П3
X9	in2,1		ПЗ3		П4
X10	in2,2		Ф		П5
X11	in2,3				П6
X12	in2,4				П7
X13	in2,5				КП
X14	in2,6				КСЛ
X15	in2,7				КСС

Таблица 25. Сигналы управления

Номер бита (команды) в ПЭВМ	Адрес бита в ОЗУ микроконтроллера	№ 1 Роботизированный комплекс	№ 2 Методическая печь	№ 3 Нагревательный колодец	№ 4 Сортировка и пакетирование
Y0	out1,0	Вв	Рв	Вверх	Тл
Y1	out1,1	Вн	Рн	Вниз	Тп
Y2	out1,2	Л	Рм	Мк	Тм
Y3	out1,3	П	Тв	Пр	Вп
Y4	out1,4	Зажим	Тн	Л	Нп
Y5	out1,5	Разжим	ЗО	Мт	Мп
Y6	out1,6	Тр1	ЗЗ		Э
Y7	out1,7	Тр2	Р1		
Y8	out2,0	Тр3	Р3		
Y9	out2,1				
Y10	out2,2				
Y11	out2,3				
Y12	out2,4				
Y13	out2,5				
Y14	out2,6				
Y15	out2,7				

Вариант 1. Роботизированный комплекс транспортировки изделий

Роботизированный комплекс (рис. 13) предназначен для переноса изделий с автоматизированных линий (транспортеров) 5 и 6 на транспортер 7.

Непосредственно робот содержит механизм подъема 1, управляемый от двигателя 2, привод выдвижения руки 3 с двигателем 4, схват 12, управляемый пневмоклапаном. Вертикальные перемещения манипулятора контролируются тремя индуктивными датчиками 9, 10, 11 (датчики К1, К2, К3), установленными прямо напротив соответствующих линий подачи заготовки. По горизонтали привод руки контролируется двумя датчиками 13 и 14 (датчики К4, К5) крайних левого и правого положений. Наличие заготовок определяется срабатыванием датчиков 15, 16 и 17 (датчики К6, К7, К8). Перемещение заготовок выполняется нереверсивными транспортерами 5, 6 и 7, при этом два первых транспортера 5 и 6 включаются для подачи заготовок влево, а транспортер 7 – направо.

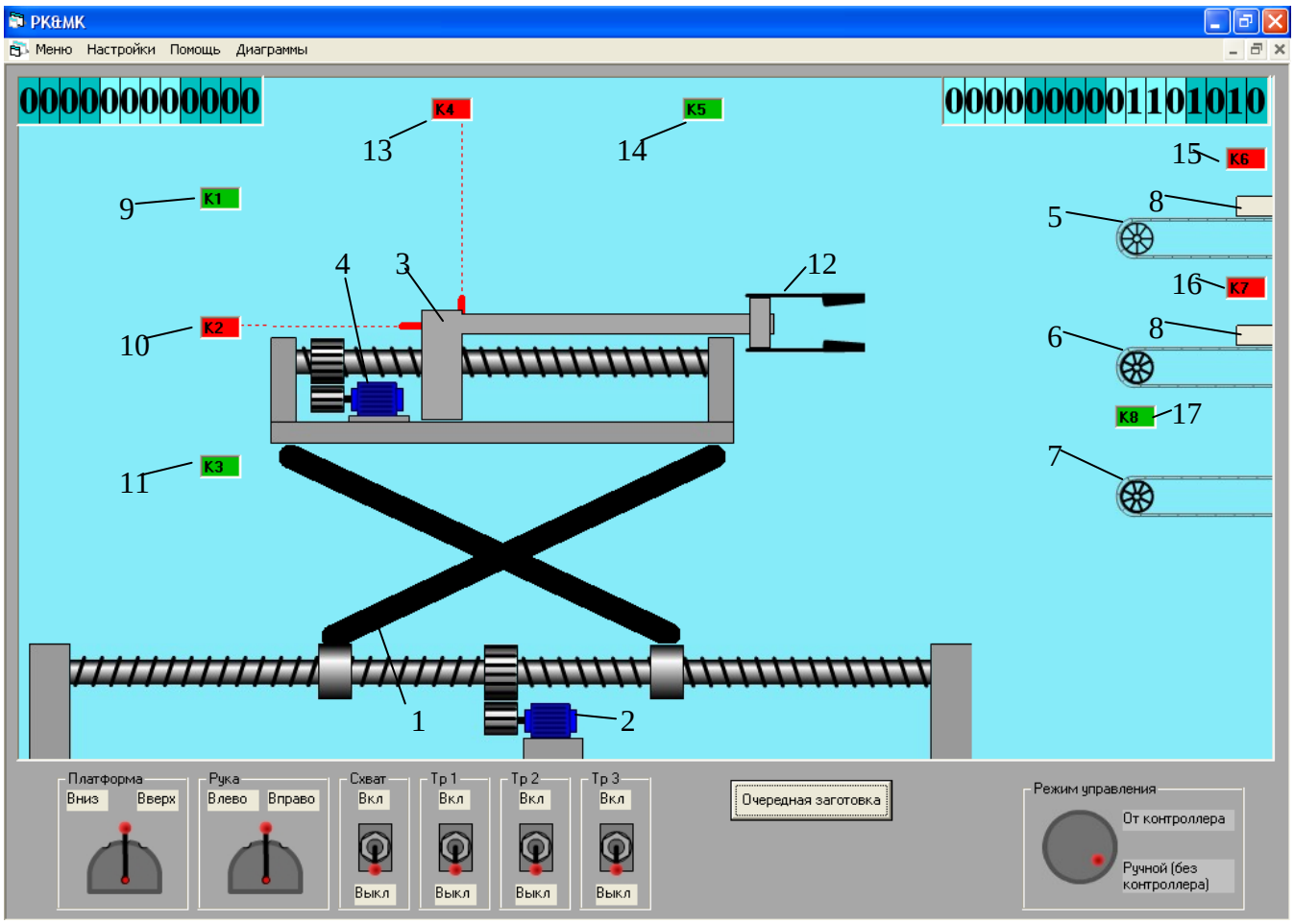


Рис. 13. Роботизированный комплекс транспортировки изделий

Цикл работы системы следующий. При нажатии кнопки «Очередная заготовка» (в нижней части экрана ПЭВМ) по закону случайных чисел появляется заготовка на подающем транспортере 5 или 6 и перемещается без участия манипулятора до срабатывания датчика К6 или К7. Система должна включить соответствующий транспортер (команды Тр1 или Тр2) до отключения того же датчика. После этого происходит перемещение манипулятора по вертикали и горизонтали к данной заготовке (исходное положение схвата предполагается разжатое) до срабатывания датчиков К1/К2 и К5, зажим схвата (команда Зажим) и отход манипулятора назад до положения К4. Если на третьем транспортере 7 отсутствует заготовка, манипулятор перемещается вниз (Вн) и вправо (П) до срабатывания датчиков К3 и К5, в противном случае система ждет освобождения этого транспортера (отсутствия сигнала датчика К8). После установки руки манипулятора около третьего транспортера 7 схват разжимается (Разжим), и при наличии сигнала датчика К8 включается транспортер Тр3 и работает до исчезновения сигнала датчика К8. Далее цикл повторяется.

Подвариант 1.1

Необходимо автоматизировать работу двух механизмов – привода подъема манипулятора и привода выдвижения руки. В ручном режиме управления манипулятор устанавливается в положение К1, К4. В автоматическом режиме по команде «Пуск» с выбранной кнопки блока управления рабочего места про-

исходит перемещение только по одной координате: по горизонтали до положения К5, по вертикали до положения К3, по горизонтали до положения К4, по вертикали до положения К1 и далее повтор. Таким образом, манипулятор описывает по часовой стрелке в пространстве прямоугольник. При нажатии кнопки «Стоп» (выбранная кнопка на блоке управления рабочего места) манипулятор устанавливается в положение К1, К4 и останавливается.

Подвариант 1.2

Необходимо автоматизировать работу двух механизмов – привода подъема манипулятора и схвата. При запуске манипулятор устанавливается в положение К2, К5, схват разжат. Затем происходит перемещение только по вертикали: вниз до положения К3 и затем зажать схват, по вертикали до положения К1 и затем схват разжать и так далее.

Подвариант 1.3

Требуется автоматизировать процесс тестового контроля всех приводов и датчиков робота.

При нажатии выбранной кнопки «Пуск» на блоке управления рабочего места схват робота должен разжаться, рука прийти в положение К4, и механизм подъема опустить руку в положение К3. Через задержку времени t = 2…3 с осуществляется подъем руки до положения К1 с остановкой на время t в положении К2. Затем рука движется в положение К5, стоит в нем в течение времени t, происходит зажим схвата и возвращение руки вновь в положение К4. Через задержку времени t происходит разжим схвата и опускание руки в положение К3.

При исправности всех датчиков и работоспособности всех механизмов на блоке управления рабочего места загорается зеленый светодиод. Если же при проверке были сбои, то через 60 с после нажатия кнопки «Пуск» загорается красный светодиод.

При составлении программы для микроконтроллера считать, что выполнение команд на включение привода подъема, горизонтального перемещения руки и команды на зажим и разжим схвата является подтверждением их работоспособности.

Вариант 2. Участок транспортировки заготовок в методическую печь для их нагрева перед прокаткой (методическая печь)

В методической печи 1 (рис. 14) осуществляется нагрев заготовок постоянных геометрических размеров до температуры проката.

Вначале толкатель 3 находится в исходном (крайнем заднем) положении ПТ0, заслонка 7 переднего окна печи закрыта (есть сигнал датчика ПЗ3), заготовка перед печью отсутствует. При появлении заготовки 6 в положении ПР0 подающего рольганга 4 (Р1) включается рольганг Р2 вперед (команда Рв) и заготовка движется к переднему окну печи. При достижении передним концом заготовки положения ПР1 осуществляется снижение скорости рольганга Р2 до ползучей скорости (есть команды Рв и Рм), с которой осуществляется движение заготовки до положения ПР2. Рольганг Р2 отключается. Если по инерции пе-

редний конец дошел до положения ПР3 или перешел его, то рольганг Р2 включается для движения назад на ползучей скорости (команды Рн и Рм), движение сразу прекращается, если заготовка уйдет из положения ПР3.

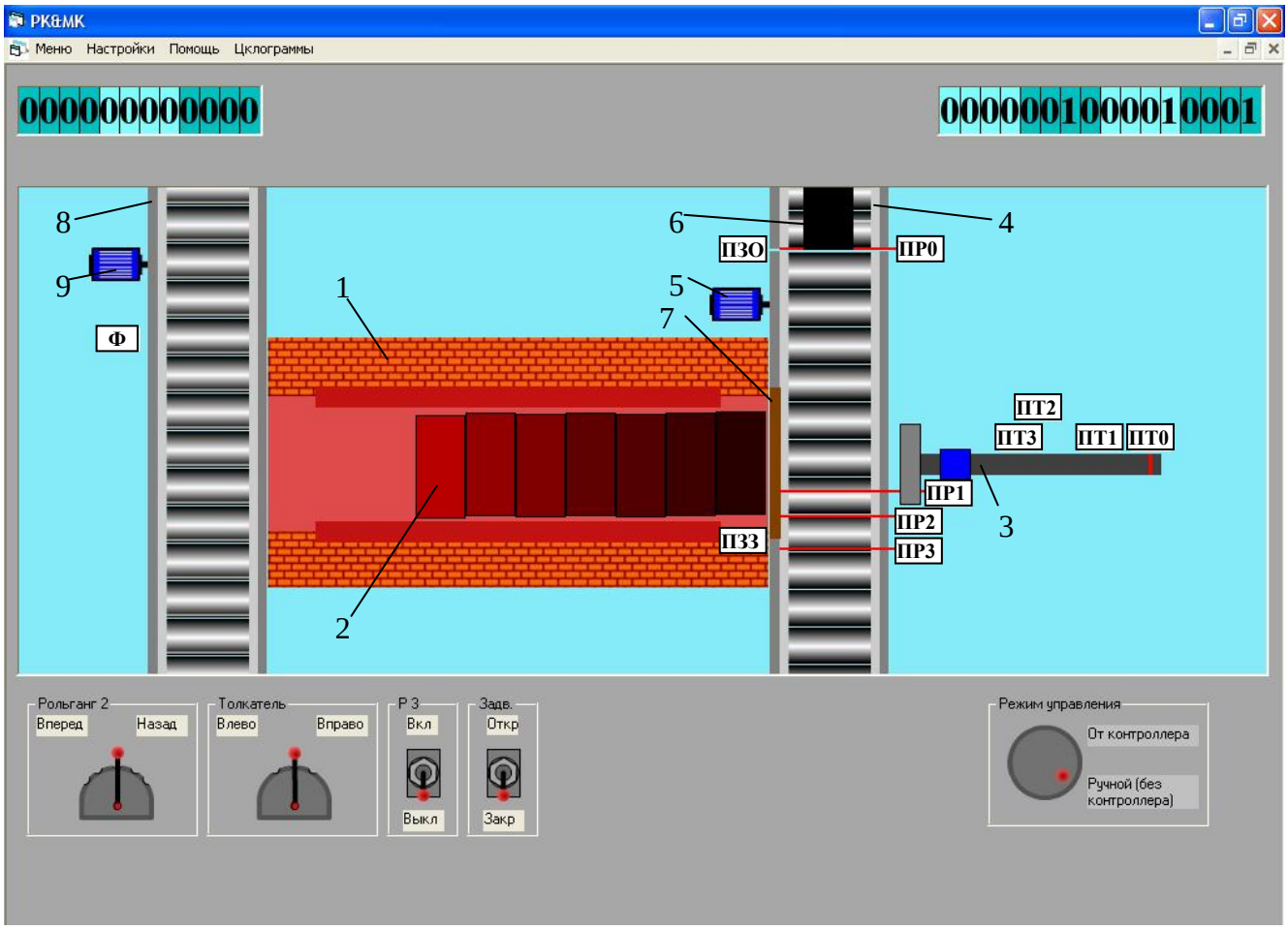


Рис. 14. Участок транспортировки заготовок в методическую печь

Заготовка стоит перед печью до прихода сигнала (запроса) на выдачу заготовки из печи с выбранной кнопки на блоке управления рабочего места. В этом случае включается толкатель вперед (команда Тв) и движется до положения ПТ1. В этом положении толкателя включается двигатель заслонки 7 на ее открытие (команда ЗО). Заслонка уходит из закрытого положения заслонки ПЗЗ и при достижении открытого положения заслонки ПЗО останавливается. Вновь включается толкатель для движения вперед. При своем движении толкатель сталкивает заготовку с рольганга и подает ее в печь 1, передвигая при этом все нагреваемые заготовки в печи. В режиме работы с полной загрузкой печи толкатель должен проделать путь, при котором крайняя заготовка выталкивается из печи и попадает на отводящий рольганг 8 (Р3). При этом засвечивается фотоэлектрический датчик Ф, при срабатывании которого включается отводящий рольганг (Р3), транспортируя заготовку к прокатному стану, а толкатель реверсируется (Тн) и движется в исходное положение. При прекращении засветки датчика Ф отводящий рольганг останавливается.

Если идет еще только первоначальная загрузка печи и при движении толкателя вперед крайняя заготовка в принципе не может быть вытолкнута на от-

водящий рольганг, то толкатель реверсируется при достижении им положения ПТ3.

Когда толкатель при своем ходе назад (команда Тн) проходит положение ПТ2, формируется команда на закрытие заслонки (ЗЗ). Она закрывается до тех пор, пока не придет в положение ПЗЗ.

При остановке толкателя в положении ПТ0 возможна подача к печи очередной заготовки.

Рольганг Р2 приводится в движение двигателем 5. Привод заслонки условно не показан. Привод отводящего рольганга приводится в движение двигателем 9. Толкатель приводится в движение двигателем, который условно не показан.

Подвариант 2.1

Необходимо автоматизировать работу двух механизмов – толкателя и задвижки. При поступлении кратковременной команды "Запрос оператора" с блока управления рабочего места электропривод толкателя включается и из исходного положения ПТ0 толкатель движется к промежуточному положению ПТ1, где останавливается. В этот момент начинает открываться заслонка, и после ее открытия толкатель вновь движется вперед. При достижении толкателем положения ПТ3 происходит реверс и он возвращается в исходное состояние ПТ0. Как только толкатель при движении назад проходит положение ПТ2, заслонка закрывается. Цикл повторяется при повторном нажатии на кнопку "Запрос оператора" с блока управления рабочего места.

Подвариант 2.2

При наличии заготовки в положении ПР0 и нажатии на кнопку "Пуск" (выбранная кнопка на блоке управления рабочего места) происходит включение рольганга Р2 и движение заготовки до положения ПР2, где и останавливается. По сигналу со следующей выбранной кнопки блока управления рабочего места осуществляется возвращение заготовки на повышенной скорости рольганга Р2 до воздействия на датчик ПР1. Происходит останов рольганга. Через 0,5 с заготовка возвращается в положение ПР2. Формируется сигнал "Конец цикла".

Подвариант 2.3

При наличии заготовки в положении ПР0 и нажатии кнопки "Пуск" (выбранная кнопка на блоке управления рабочего места) включается подающий рольганг. При достижении заготовкой положения ПР2 рольганг останавливается, заслонка открывается и включается привод толкателя. При достижении толкателем положения ПТ1 проверяется условие полного открытия заслонки (т.е. достижения положения ПЗО). Если заслонка полностью открыта, то толкатель продолжает движение вперед до положения ПТ3. Если же заслонка еще не открыта, то толкатель останавливается, а затем после полного открытия заслонки продолжает движение до положения ПТ3.

В положении ПТ3 привод толкателя реверсируется и толкатель возвращается в исходное положение ПТ0. При движении назад и проходе толкателем положения ПТ1 подается команда на закрытие заслонки.

Вариант 3. Участок нагревательного колодца обжимного прокатного стана

Нагрев слитков перед прокаткой на обжимном прокатном стане осуществляется в нагревательных колодцах, отапливаемых смесью доменного и коксового газов или природным газом. Колодец закрывается крышкой. Для открытия и закрытия колодца крышкой существует напольно-крышечная машина (рис. 15), включающая два механизма: механизм подъема – опускания крышки и механизм перемещения тележки, на которой располагается крышка с механизмом подъема – опускания.

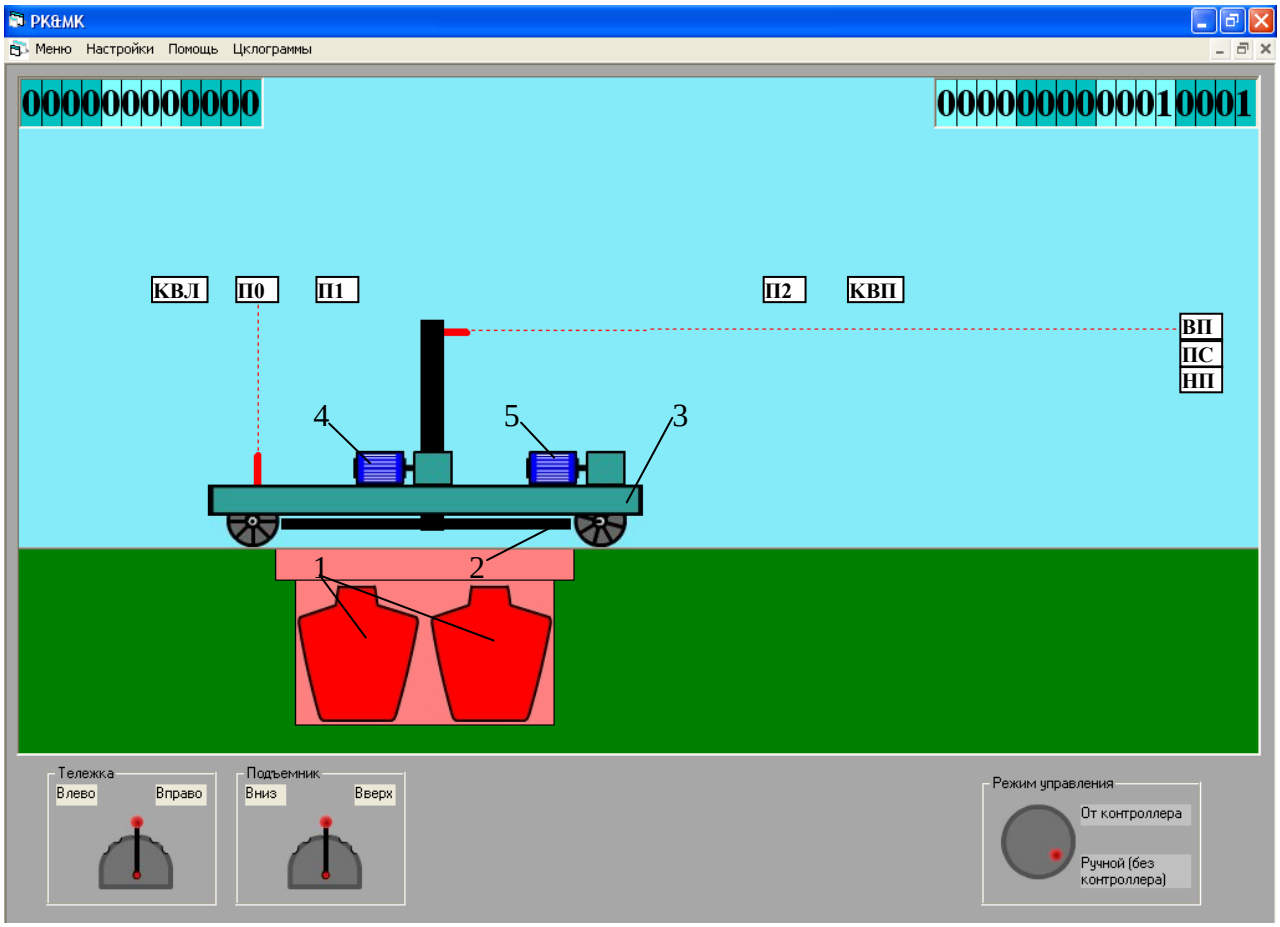


Рис. 15. Механизм управления крышкой нагревательных колодцев

При нагреве слитков 1 до температуры 1200 °С к колодцу подаются слитковоз и клещевой кран (на рисунке не показаны). По сигналу "Открыть" (с выбранной кнопки блока управления рабочего места лабораторного комплекса) включается (Вверх) двигатель 4 (Д1) механизма подъема крышки колодца, установленный на тележке 3. Двигатель Д1 через механическую передачу осуществляет подъем крышки 2 до верхнего положения ВП, контролируемого соответствующим датчиком. Затем включается двигатель 5 (Д2) и тележка, приводимая в движение указанным двигателем, движется в позицию ожидания П2

(движение вправо (Пр)) и стоит там до тех пор, пока с блока управления рабочего места не поступит сигнал "Закрыть". В этом случае тележка движется влево (команда Л). При движении влево в положении П1 осуществляется снижение скорости движения тележки до ползучей скорости (команда Мт), с которой тележка движется до положения П0. В положении П0 происходит останов тележки и осуществляется движение крышки вниз (команда Вниз). При наличии сигнала с датчика ПС при движении вниз происходит снижение скорости опускания крышки до ползучей (команда Мк), с которой происходит затем движение до полного закрытия колодца. Нижнее положение крышки контролируется датчиком НП (закрыт колодец).

При загрузке нагревательного колодца слитками все операции осуществляются аналогично, только вместо слитковоза подается состав со слитками.

В программе ПЭВМ предусмотрена остановка движения тележки при достижении ею крайних положений КВЛ и КВП (конечный выключатель левый и конечный выключатель правый соответственно) независимо от режима работы напольно-крышечной машины.

Подвариант 3.1

Автоматизировать работу механизма управления крышкой нагревательных колодцев. При подаче сигнала "Пуск" (с выбранной кнопки блока управления рабочего места) открывается крышка колодца и тележка движется вправо до П2, где через 1 с происходит реверс. В положении П1 тележка переходит на малую скорость и возвращается в исходное состояние П0. Крышка колодца закрывается. При достижении крышкой положения НП цикл заканчивается.

Подвариант 3.2

При первом нажатии на кнопку "Пуск" открывается крышка нагревательного колодца до положения ВП, где останавливается на 2 с и возвращается в исходное состояние. Затем крышка доходит до положения ПС, где останавливается на 3 с и возвращается в исходное состояние. Движение "Вниз" от положения ПС к положению НП осуществляется на медленной скорости. Далее цикл непрерывно повторяется до нажатия на кнопку "Стоп" с блока управления рабочего места.

Подвариант 3.3

В режиме ручного управления «Без контроллера» поставить крышку колодца в крайнее верхнее положение ВП. Установить режим управления «От контроллера». После первого нажатия на выбранную кнопку блока управления рабочего места (условно кнопка "Пуск") тележка движется вправо до П2, где стоит 2 с. После окончания выдержки времени тележка перемещается влево до положения П1, переходит на ползучую скорость и подходит к П0, где останавливается до очередного нажатия кнопки "Пуск". Если в процессе работы нажималась кнопка «Стоп», то движение тележки прекращается, но если затем нажать кнопку «Пуск», то движение тележки продолжается с места остановки.

Вариант 4. Участок сортировки и пакетирования годных и бракованных листов металла

Обработанные и нарезанные на мерные длины стальные листы металла проходят через устройство контроля качества, которое выдает информацию о качестве листа. При качественном (годном) листе выдается сигнал годности Г, при бракованном листе выдается сигнал брака Б (рис. 16).

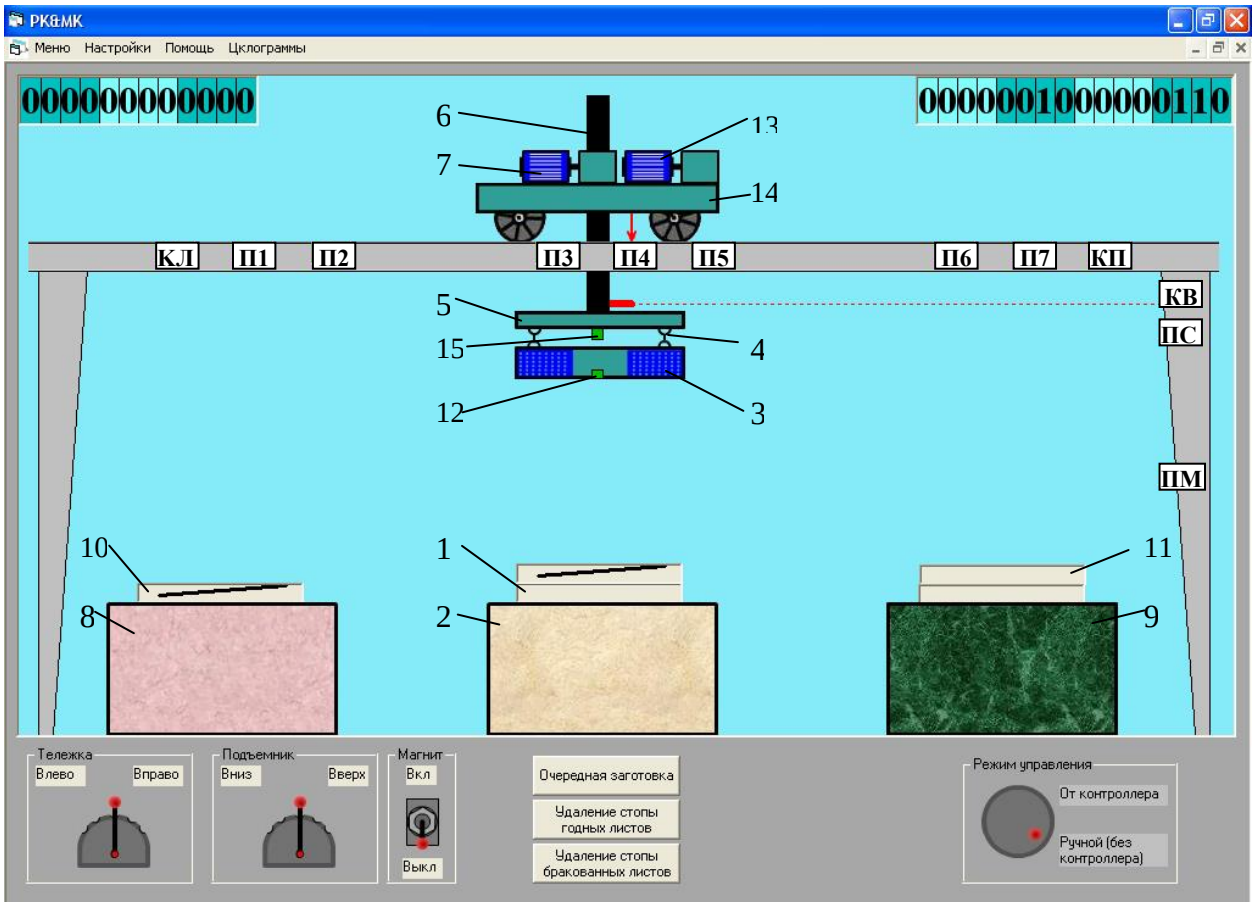


Рис. 16. Участок сортировки и пакетирования листов

Лист 1 подается на стол 2 раскладывателя при нажатии на кнопку «Очередная заготовка» в нижней части экрана монитора. Захват и перенос листа со стола осуществляется электромагнитом 3, подвешенным цепями 4 к траверсе 5. Если лист захвачен (есть сигнал датчика касания листа 12 (КСЛ)), то включается двигатель 7 (Д1) подъемника и осуществляется подъем (Вп) штанги 6, а следовательно, и электромагнита с листом. Подъем происходит до срабатывания датчика крайнего верхнего положения КВ. После отключения двигателя Д1 включается двигатель 13 (Д2) перемещения тележки 14 подъемника. Тележка движется влево (Тл) к столу 8, если лист бракованный, или вправо (Тп) к столу 9, если лист годный. Тележка разгоняется до заданной скорости. При приближении к столу пакетирования по сигналу датчика П2 (или П6) происходит снижение скорости движения тележки до ползучей скорости (Тм) и по сигналу датчика П1 (или П7) происходит отключение двигателя Д2.

Подъемник опускается (Нп) для укладывания листа в стопу. При соприкосновении листа со стопой его опускание с электромагнитом прекращается, но

опускание траверсы 5 продолжается. На траверсе закреплен датчик касания стопы 15 (КСС). При срабатывании этого датчика прекращается опускание подъемника и снимается питание с электромагнита. Подъемник возвращается в положение КВ.

Вновь включается привод тележки раскладывателя и тележка движется в положение П4 над столом 2. По сигналу датчиков П3 или П5 (в зависимости от направления подхода к положению П4) происходит снижение скорости движения тележки до ползучей скорости (Тм). По сигналу датчика П4 происходит останов тележки.

При поступлении очередного листа и наличии сигнала о его качестве из положения КВ подъемник опускается (Нп) до положения ПМ, в котором осуществляется снижение скорости подъемника до ползучей скорости (Мп), с которой подъемник опускается до касания листа (до срабатывания датчика КСЛ). Происходит включение электромагнита (Э) и цикл работы повторяется.

Необходимо предусмотреть формирование сигналов, выдаваемых на светодиоды блока управления рабочего места лабораторного комплекса о переполнении стоп бракованных 10 и годных 11 листов. Максимальная высота стоп годных и бракованных листов одинаковая и контролируется датчиком переполнения ПС.

В программе ПЭВМ предусмотрено случайное генерирование информации годного или бракованного листа. Очередной лист на столе 2 появляется при нажатии кнопки «Очередная заготовка» в нижней части экрана монитора. Если нет ни сигнала годного листа, ни сигнала бракованного листа, то электромагнит должен стоять в исходном положении.

Уборка стоп бракованных и годных листов осуществляется нажатием соответственно кнопок «Удаление стопы бракованных листов» и «Удаление стопы годных листов» в нижней части экрана монитора.

Начало автоматической работы возможно при наличии сигнала КВ и расположении тележки в положении П4. В указанные положения механизмы устанавливаются при вызове программы рассматриваемого варианта или приводятся в указанные положения в ручном режиме управления «Без контроллера».

В программе ПЭВМ предусмотрена остановка движения подъемника при достижении им положения КВ и остановка движения тележки при достижении ею крайних положений КЛ и КП независимо от режима работы установки.

Подвариант 4.1

Требуется автоматизировать процесс перемещений тележки раскладывателя. Если траверса не находится в верхнем положении, то необходимо предусмотреть подъем траверсы в ручном режиме работы. Исходное положение тележки П4. При каждом нечетном нажатии на кнопку «Пуск» (выбранная кнопка на блоке управления рабочего места) тележка из положения П4 движется в положение П2, стоит в нем в течение 2 с и возвращается в положение П4. При каждом четном нажатии на кнопку «Пуск» тележка из положения П4 движется в положение П7, стоит в нем в течение 3 с и возвращается в положение П4.

В момент достижения тележкой положения П4 (при каждом четном нажатии на кнопку «Пуск») включается сигнал «Конец цикла», который выключается очередным сигналом «Пуск» или подачей сигнала «Сброс» с блока управления рабочего места.

Подвариант 4.2

Требуется автоматизировать процесс перемещения двух листов со стола раскладывателя (см. рис. 16) на правый стол, независимо от качества листа.

Из исходного состояния при наличии листов на столе (тележка находится в положении П4, подъемник в верхнем положении КВ) по команде «Пуск» (выбранная кнопка на блоке управления рабочего места) подъемник опускается вниз на ползучей скорости до момента срабатывания датчика касания листа КСЛ. При срабатывании этого датчика выключается опускание подъемника и включается питание электромагнита, тем самым происходит захват листа. Далее на повышенной скорости осуществляется подъем штанги и электромагнита с листом до верхнего положения КВ. Затем тележка на ползучей скорости перемещается из положения П4 в положение П7, где останавливается.

Подъемник опускает лист на стол 9 и оставляет его там (опускание осуществляется до срабатывания датчика КСС). Затем подъемник возвращается в исходное состояние КВ, а тележка – в положение П4.

Затем совершается еще один цикл перекладывания листа со стола 2 на стол 9. При возвращении раскладывателя в исходное состояние загорается сигнал «Конец цикла». Повторение цикла – при повторном нажатии кнопки «Пуск».

Подвариант 4.3

Необходимо произвести автоматизированный тестовый контроль работы всех механизмов и датчиков раскладывателя. Оценку работы всех датчиков и механизмов визуально осуществляет оператор. Начало контроля осуществляется после нажатия выбранной кнопки на блоке управления рабочего места. В конце контроля должна загореться сигнальная лампа «Конец контроля». При работе оборудования не должны создаваться аварийные ситуации.

Библиографический список

1. Евстифеев А.В. Микроконтроллеры AVR семейства Mega : Рук. пользователя. – М. : Издательский дом «Додэка-XXI», 2007. – 592 с.

2. Лабораторный комплекс «Микроконтроллеры и автоматизация». Техн. описание. – Челябинск : НПП «Учебная техника – Профи», 2009. – 20 с.

3. Лабораторный комплекс «Микроконтроллеры и автоматизация» : Метод. указания к проведению лаб. работ. – Челябинск : НПП «Учебная техника – Профи», 2009. – 98 с.

4. Программирование на ассемблере для AVR-микроконтроллеров : Лаб. практикум по основам микропроцессорной техники / А. Ю. Бальзамов. – Саранск : Изд-во Мордов. ун-та, 2012. – 108 с.