ВВЕДЕНИЕ

Целью настоящих методических указаний является повышение уровня самостоятельной работы студентов при выполнении лабораторных работ по курсу «Микропроцессоры и микропроцессорные системы управления» и изучению 8 разрядных процессоров AVR.

Методические указания составлены в соответствии с программой курса «Микропроцессоры и микропроцессорные системы управления». В них содержится информация по пяти лабораторным работам:

- основы программирования 8-разрядных процессоров семейства ATmega;
- команды управления программой 8-разрядных процессоров семейства ATmega;
- реализация системы управления на 8-разрядных микроконтроллерах семейства ATmega8515;
- реализация системы управления на 8-разрядных микроконтроллерах семейства ATmega8515: работа систем в реальном масштабе времени;
- реализация системы управления на 8-разрядных микроконтроллерах семейства ATmega8515.

В каждой лабораторной работе содержатся краткое описание, цель работы, порядок выполнения работы, отчетность, вопросы по зачетам и список литературы.

К выполнению лабораторных работ студенты могут приступить только после изучения соответствующего раздела курса, используя курс лекций, техническую литературу и литературу по 8 разрядным процессорам AVR, приведенную в конце методических указаний.

Разрешение на выполнение работ дает преподаватель, после того, как убедится в наличии у студентов знаний по выполнению и оформлению соответствующей работы.

В каждом отчете по лабораторной работе должны быть сделаны выводы.

Защита лабораторных работ производится индивидуально по теоретическому и практическому материалу. В случае несоответствия экспериментальных данных теоретически расчетным, студенты в ходе зачета должны дать соответствующие пояснения.

Настоящие методические указания помогут студентам подготовиться к лабораторным занятиям, повысить качество отчета и ее защиту.

Настоящие методические указания к проведению лабораторных работ по курсу «Микропроцессоры и микропроцессорные системы управления» могут быть использованы для других специальностей радиотехнического профиля.

ЛАБОРАТОРНАЯ РАБОТА №1

Основы программирования 8-разрядных процессоров семейства ATmega

Цель работы:

Знакомство с интегрированной средой программирования AVR Studio4, изучение архитектуры и регистровой структуры процессора семейства ATmega.

Знакомство с реализуемыми процессором способами адресации и командами пересылки данных.

Знакомство с арифметическими и логическими командами, а так же операциями сравнения.

Используемое оборудование:

- персональная ЭВМ, совместимая с IBM PC.

Используемое программное обеспечение:

- операционная система Windows XP;
- интегрированная среда программирования AVR Studio4.

1. Краткое описание работы

1.1. Программирование и отладка процессоров семейства ATmega с помощью интегрированной среде программирования AVR Studio4

На рисунке 1 приведен вид рабочего экрана AVR Studio4 в режиме симулятора с открытыми окнами ассемблера и аппаратных ресурсов МК ATmega 8515.

Рис. 1. Вид экрана монитора при работе
с интегрированной средой программирования AVR Studio4

Используемая в лабораторном практикуме интегрированная среда программирования AVR Studio4 которая обеспечивает следующие режимы: симулятор; схемный эмулятор; программатор.

Симуляторы – это программы, которые выполняют откомпилированный программный код в компьютере системы разработки регистровой модели микроконтроллера (МК).

Специальная схема, реализующая интерфейс с МК в реальном масштабе времени, называется схемный эмулятор. Последний инструмент разработчика – программатор памяти программ МК.

Интегрированная среда программирования AVR Studio4 имеет следующие режимы меню:

File – выполнение операций с файлами;

Project – выполнение операций с проектом;

Build – отладка программы и запуск;

Edit – редактирование проекта или файла;

View – редактирование интерфейса программы;

Tools – настройка отладчика согласно требованиям пользователя;

Debug – выполнение проекта в режиме отладки;

Window – управление форматом и содержанием информации на экране;

Help – помощь.

Для создания нового проекта программы необходимо выбрать в меню Project → New Project, мастер создания проекта откроет окно, как показано на рис. 2.

На данном этапе вы задайте тип (расширение) файла программы, которую вы хотите создать, а также имя файла и адрес, по которому он будет храниться.

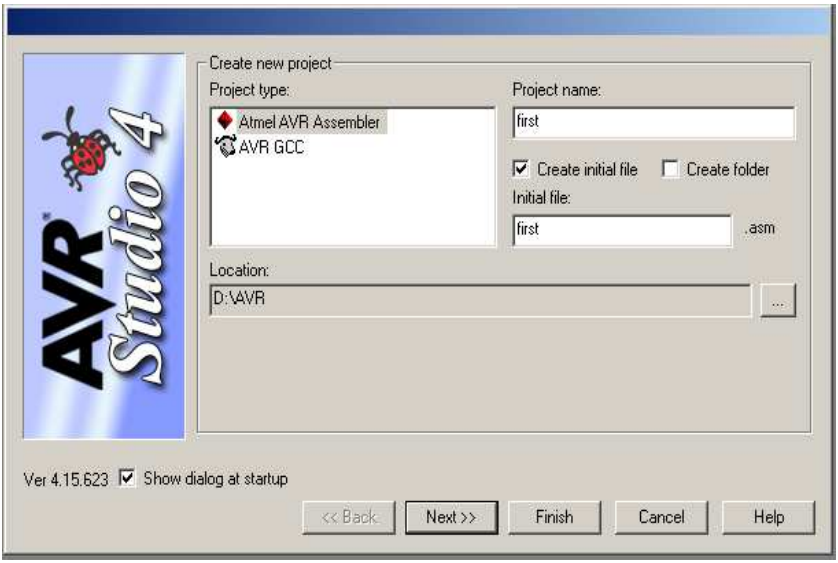

Рис.2. Мастер создания нового проекта

Для этого вам необходимо сделать следующее (рис. 2):

- AVR Studio необходимо сообщить на каком языке писать программу: Atmel AVR assembler – язык программирования ассемблера или AVR GCC – язык программирования C;

- далее необходимо задать имя вашего проекта и место его расположения;

- когда выполнены данные действия нажать «Next».

Программное обеспечение AVR Studio 4 работает с широким спектром эмуляторов и отладчиков. На данном этапе следует воспользоваться встроенным симулятором и разработать программу для устройства ATmega8515(рис 3).

На последнем этапе проверяем все параметры ещё раз, и нажимаем Finish для завершения создания проекта и переходим к файлу для программирования на ассемблере.

Для отладки и запуска программы в меню Build, Build and Run запускаем программу. Далее в пошаговом режиме Step into (F11) или Auto Step в меню Debug, выполняем программу (рис.4).

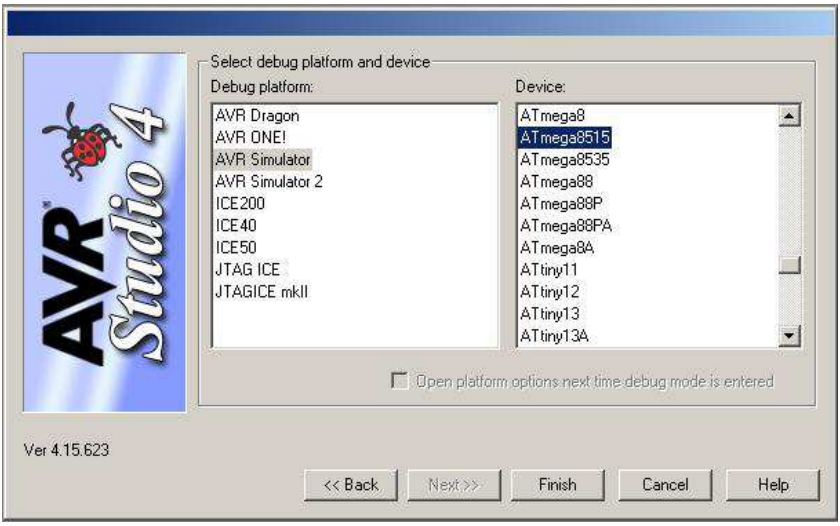

Рис.3. Выбор платформы и устройства для разработки программы

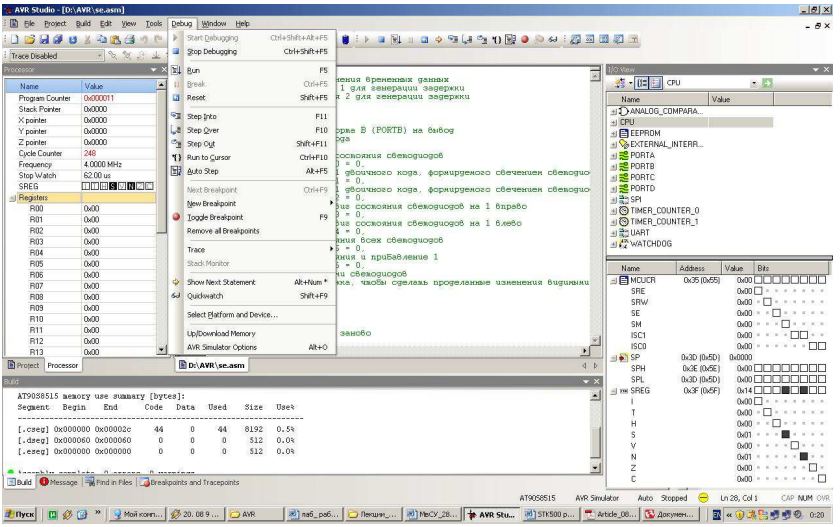

Рис.4. Отладка и запуск программы

Отладочная плата STK500 – завершенный стартовый набор и система проектирования для AVR флеш МК корпорации Atmel.

Отличительные особенности:

- совместимость с программой AVR Studio;
- связь с компьютером через интерфейс RS-232 для программирования и управления;
- стабилизированный источник питания с входом 10-15В
- 8-выв.,20-выв, 28-выв.,40-выв. Панели для установки DIP-корпусов AVR – МК;
- поддержка параллельного и последовательного программирования повышенным напряжением всех AVR – МК;
- последовательное внутрисистемное программирование (ISP) всех AVR – устройств;
- внутрисистемный программатор для программирования МК непосредственно в целевом приложении;
- перепрограммирование AVR – МК;
- 8 кнопок общего назначения;
- 8 светодиодов общего назначения;
- все порты ввода – вывода выведены на штырьки разъема;
- дополнительный порт RS-232 общего назначения;
- разъемы расширения для подключения внешних модулей и областей для макетирования;
- встроенная флеш-память DataFlash емкостью 2Мбит для энергонезависимого хранения данных.

1.2. Общие сведения микроконтроллеров AVR фирмы Atmel семейства mega

Как и все МК AVR фирмы Atmel, МК семейства Mega являются 8-битными МК, предназначенными для использования во встраиваемых приложениях. Они изготавливаются по малопо-

требляющей КМОП-технологии, которая в сочетании с усовершенствованной RISC-архитектурой позволяет достичь наилучшего соотношения стоимость, быстродействие, энергопотребление. МК описываемого семейства являются наиболее развитыми представителями МК AVR общего применения.

1.2.1 Отличительные особенности микроконтроллеров AVR семейства Mega

К особенностям МК AVR семейства Mega можно отнести:

- 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;
- 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 детектора пониженного напряжения питания;

- возможность программного снижения частоты тактового генератора (не во всех моделях).

1.2.2 Архитектура и характеристики процессора семейства mega

Ядро МК AVR семейства Mega выполнено по усовершенствованной RISC-архитектуре (рис. 5), в которой используется ряд решений, направленных на повышение быстродействия МК.

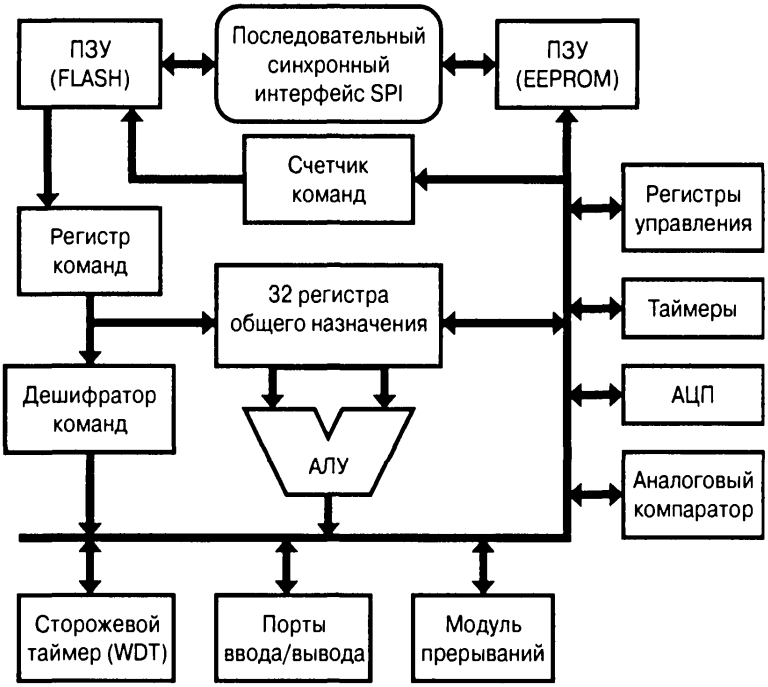

Рис.5 Архитектура процессора микроконтроллера AVR

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 занимает одну ячейку памяти программ.

В МК 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.

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 такая архитектура позволяет достичь производительности в 1 млн. оп. в сек на каждый 1 МГц тактовой частоты.

Основными характеристиками процессора МК AVR семейства Mega являются:

- полностью статическая архитектура, минимальная тактовая частота равна нулю;
- арифметико-логическое устройство (АЛУ) подключено непосредственно к регистрам общего назначения (32 регистра);
- большинство команд выполняются за один период тактового сигнала;
- векторная система прерываний, поддержка очереди прерываний;
- большое число источников прерываний (до 45 внутренних и до 32 внешних);
- наличие аппаратного умножителя.

1.2.3 Характеристики подсистемы ввода/вывода микроконтроллеров AVR семейства Mega

Подсистема ввода/вывода МК AVR семейства Mega имеет следующие особенности:

- программное конфигурирование и выбор портов ввода/вывода;
- выводы могут быть запрограммированы как входные или как выходные независимо друг от друга;
- входные буферы с триггером Шмитта на всех выводах;
- на всех входах имеются индивидуально отключаемые внутренние подтягивающие резисторы сопротивлением 20...50 кОм.

1.2.4 Периферийные устройства микроконтроллеров AVR семейства Mega

МК семейства Mega имеют 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;
- сторожевой таймер;
- одно- и двухканальные генераторы 8-битного ШИМ-сигнала;
- двух- и трехканальные генераторы ШИМ-сигнала регулируемой разрядности. 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;

- от одного до четырех полнодуплексных универсальных синхронных/асинхронных приемо-передатчиков (USART);

- универсальный последовательный интерфейс USI, который может использоваться в качестве интерфейса SPI или I^2C.

1.3. Цоколевка и описание выводов микроконтроллера ATmega8515

На рис.6. представлено расположение выводов и назначение портов ввода-вывода МК ATmega8515 в PDIP корпусе табл.1.

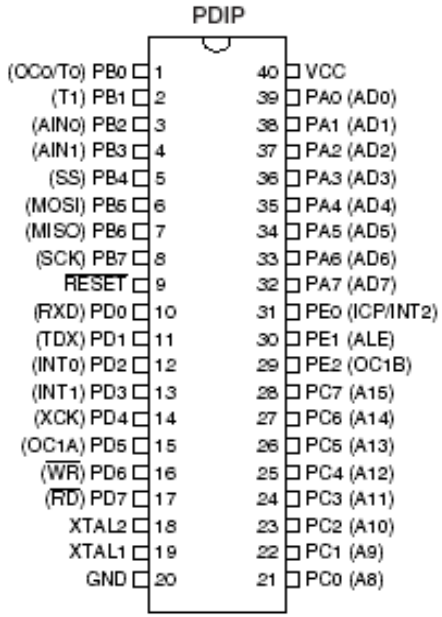

Рис. 6 Расположение выводов МК ATmega8515

МК ATmega8515 выпускается в следующих корпусах: 40-выв. PDIP; 44-выв. TQFP; 44-выв. PLCC и 44-выв. MLF.

Напряжение питания: 5.5В для ATmega8515L; 4.5 - 5.5В для ATmega8515;

Рабочая частота: 0 - 8 МГц для ATmega8515L; 0 - 16 МГц для ATmega8515.

Таблица 1.

Назначение портов ввода-вывода МК ATmega8515

Обозначение	Номер вывода			Тип вывода	Описание
	DIP	TQFP MLF	PLCC		
XTAL1	19	15	21	I	Вход тактового генератора
XTAL2	18	14	20	O	Выход тактового генератора
RESET	9	4	10	I	Вход сброса
Порт А. 8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами					
PA0 (AD0)	39	37	43	I/O	0-й бит порта А 0-й бит мультиплексированной ША/ШД для внешнего ОЗУ
PA1 (AD1)	38	36	42	I/O	1-й бит порта А 1-й бит мультиплексированной ША/ШД для внешнего ОЗУ
PA2 (AD2)	37	35	41	I/O	2-й бит порта А 2-й бит мультиплексированной ША/ШД для внешнего ОЗУ
PA3 (AD3)	36	34	40	I/O	3-й бит порта А 3-й бит мультиплексированной ША/ШД для внешнего ОЗУ
PA4 (AD4)	35	33	39	I/O	4-й бит порта А 4-й бит мультиплексированной ША/ШД для внешнего ОЗУ
PA5 (AD5)	34	32	38	I/O	5-й бит порта А 5-й бит мультиплексированной ША/ШД для внешнего ОЗУ
PA6 (AD6)	33	31	37	I/O	6-й бит порта А 6-й бит мультиплексированной ША/ШД для внешнего ОЗУ

PA7 (AD7)	32	30	36	I/O	7-й бит порта A 7-й бит мультиплексированной ША/ШД для внешнего ОЗУ
Порт B. 8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами.					
PB (T0/OC0)	1	40	2	I/O	0-й бит порта B Вход внешнего тактового сигнала таймера/счетчика T0 Выход таймера/счетчика T0
PBI (T1)	2	41	3	I/O	1-й бит порта B Вход внешнего тактового сигнала таймера/счетчика T1
PB2 (AIN0)	3	42	4	I/O	2-й бит порта B Неинвертирующий вход компаратора
PBI3(AIN1)	4	43	5	I/O	3-й бит порта B Инвертирующий вход компаратора
PB4 (SS)	5	44	6	I/O	4-й бит порта B Выбор Slave-устройства на шине SPI
PB5 (MOSI)	6	1	7	I/O	5-й бит порта B Выход (Master) или вход (Slave) тактового сигнала модуля SPI
PB6 (MISO)	7	2	8	I/O	6-й бит порта B Вход (Master) или выход (Slave) данных модуля SPI
PB7 (SCK)	8	3	9	I/O	7-й бит порта B Выход (Master) или вход (Slave) тактового сигнала модуля SPI
Порт C. 8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами.					
PC0 (A8)	21	18	24	I/O	0-й бит порта C 8-й бит ША для внешнего ОЗУ
PC1 (A9)	22	19	25	I/O	1-й бит порта C 9-й бит ША для внешнего ОЗУ
PC2 (A10)	23	20	26	I/O	2-й бит порта C 10-й бит ША для внешнего ОЗУ
PC3 (A11)	24	21	27	I/O	3-й бит порта C 11-й бит ША для внешнего ОЗУ
PC4 (A12)	25	22	28	I/O	4-й бит порта C 12-й бит ША для внешнего ОЗУ

PC5 (A13)	26	23	29	I/O	5-й бит порта C 13-й бит ША для внешнего ОЗУ
PC6 (A14)	27	24	30	I/O	6-й бит порта C 14-й бит ША для внешнего ОЗУ
PC7 (A15)	28	25	31	I/O	7-й бит порта C 15-й бит ША для внешнего ОЗУ
Порт D. 8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами.					
PD0 (RXD)	10	5	11	I/O	0-й бит порта D Вход USART
PD1 (TXD)	11	7	13	I/O	1-й бит порта D Выход USART
PD2 (INT0)	12	8	14	I/O	2-й бит порта D Вход внешнего прерывания
PD3 (INT1)	13	9	15	I/O	3-й бит порта D Вход внешнего прерывания
PD4 (XCK)	14	10	16	I/O	4-й бит порта D Вход/Выход внешнего тактового сигнала USART
PD5 (OC1A)	15	11	17	I/O	5-й бит порта D Выход A таймера/счетчика T1
PD6 (WR)	16	12	18	I/O	6-й бит порта D Строб записи во внешнее ОЗУ
PD7 (RD)	17	13	19	I/O	7-й бит порта D Строб чтения внешнего ОЗУ
Порт E. 3-битный порт ввода/вывода с внутренними подтягивающими резисторами					
PE0 (ICP/INT2)	31	29	35	I/O	0-й бит порта E Вход захвата таймера/счетчика T1 Вход внешнего прерывания
PE1 (ALE)	30	27	33	I/O	1-й бит порта E Строб адреса внешнего ОЗУ
PE2(OC1B)	29	26	32	I/O	2-й бит порта E Выход B таймера/счетчика T1
VCC	40	38	44	P	Вывод источника питания
GND	20	26	22	P	Общий вывод
NC	-	6, 17, 28,39	1,12,23,24	-	Не используются

Параллельный порт ввода-вывода (Port P) предназначен для ввода и вывода данных. МК семейства AVR имеют от одного до шести портов и для передачи данных предназначены следующие регистры:

Параллельный порт ввода-вывода (Port D):

- PIND – Выводы порта D;
- DDRD – Регистр направления передачи данных порта;
- PORTD – Регистр данных порта D.

Параллельный порт ввода-вывода (Port C):

- PINC – Выводы порта C;
- DDRC – Регистр направления передачи данных порта C;
- PORTC – Регистр данных порта C.

Параллельный порт ввода-вывода (Port B):

- PINB – Выводы порта B;
- DDRB – Регистр направления передачи данных порта B;
- PORTB – Регистр данных порта B;

Параллельный порт ввода-вывода (Port A):

- PINA – Выводы порта A;
- DDRA – Регистр направления передачи данных порта A;
- PORTA – Регистр данных порта A;

Процессор формирует адрес очередной команды, выбирает команду из памяти и организует ее выполнение. Код команды имеет тип данных слово (16 бит) или два слова.

В счетчике команд адрес очередной команды формируется путем добавления 1 к числу, код которого хранится в счетчике команд. При пуске и перезапуске МК в счетчик команд заносится код числа 0 и первая команда выбирается из FlashROM по адресу 0.

Все регистры общего назначения (РОН) объединены в регистровый файл быстрого доступа. В МК 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. Такое решение (в сочетании с конвейерной обработкой) позволяет АЛУ выполнять одну операцию (извлечение операндов из регистрового файла, выполнение

команды и запись результата обратно в регистровый файл) за один такт.

Всем регистрам РОН присвоены имена R0, R1, ..., R31. В некоторых операциях в АЛУ могут участвовать лишь регистры со старшими номерами (от R16 до R31). Регистры с именами от R24 до R31 могут образовывать пары, используемые для хранения слов, при этом регистр с четным номером хранит младший байт, а регистр с нечетным номером – старший байт. Регистр управления МК – MCUCR.

Паре регистров R26, R27 присвоено имя X, паре регистров R28, R29 – имя Y, паре регистров R30, R31 – имя Z. Эти пары регистров используются для хранения адресов при обращениях к памяти с косвенной адресацией.

Последние 6 регистров файла могут также объединяться в три 16-битных регистра, используемых в качестве указателей при косвенной адресации памяти данных.

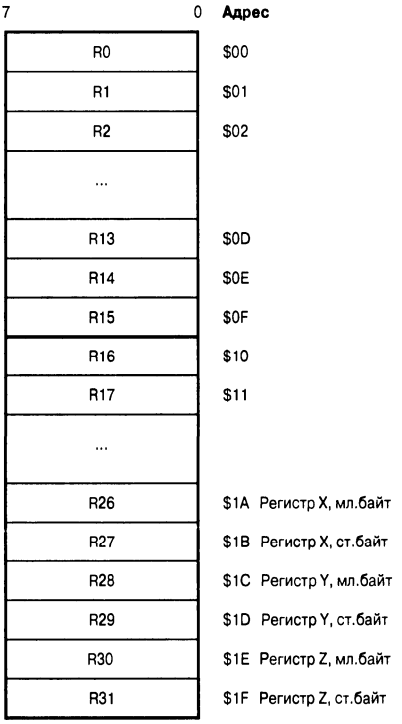

Рис.7 Регистры РОН МК ATmega8515

Каждый регистр файла имеет свой собственный адрес в пространстве памяти данных (рис.7). Поэтому к ним можно обращаться двумя способами – как к регист-

рам и как к памяти, несмотря на то, что физически эти регистры не являются ячейками памяти данных. Такое решение является еще одной отличительной особенностью архитектуры AVR, повышающей эффективность работы МК и его производительность.

Все регистры ввода/вывода (РВВ) условно можно разделить на две группы: служебные регистры МК и регистры, относящиеся к конкретным периферийным устройствам.

Во всех МК AVR регистры ввода/вывода располагаются в так называемом пространстве ввода/вывода размером 64 байта. В большинстве моделей семейства Mega имеется также пространство дополнительных регистров ввода/вывода размером 150 или 416 байт. Внедрение дополнительных РВВ связано с тем, что для поддержки всех периферийных устройств, имеющихся в этих моделях обычных 64-х РВВ недостаточно.

В таблице 2 при указании адресов РВВ в скобках указываются соответствующие им адреса ячеек памяти данных. Соответственно, если адрес регистра указывается только в скобках, этот регистр расположен в пространстве дополнительных РВВ. Если адрес в таблице не указан, это означает, что для данной модели он зарезервирован, и запись по этому адресу запрещена.

Ну и последнее - периферия, или регистры ввода-вывода (I/O). Можно прочитать данные из I/O в регистр общего назначения и записать из регистра общего назначения в I/O. Кроме этого, у части регистров ввода-вывода, а точнее - у тех, чей адрес не превышает 0x1F, возможна установка отдельных бит в состояние 0 или 1.

Таблица 2

Регистры ввода/вывода моделей ATmega 8515

Название	Адрес	Функция
SREG	$3F($5F)	Регистр состояния
SPH	$3E ($5E)	Указатель стека, старший байт
SPL	$3D ($5D)	Указатель стека, старший байт
GICR	$3B ($5B)	Общий регистр управления прерываниями

Название	Адрес	Функция
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
EMCUCR	$36 ($56)	Дополнительный регистр управления микроконтроллера
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
OCR0	$31 ($51)	Регистр совпадения таймера/счетчика T0
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
OCRIAH	$2B ($4B)	Регистр A совпадения таймера/счетчика T1, старший байт
OCRIAL	$2A ($4A)	Регистр A совпадения таймера/счетчика T1, младший байт
OCRIBH	$29 ($49)	Регистр B совпадения таймера/счетчика T1, старший байт
OCRIBL	$28 ($48)	Регистр B совпадения таймера/счетчика T1, младший байт
ICR1H	$25 ($45)	Регистр захвата таймера/счетчика T1, старший байт
ICR1L	$24 ($44)	Регистр захвата таймера/счетчика T1, младший байт
WDTCR	$21 ($41)	Регистр управления сторожевого таймера
UBRRH	$20 ($40)	Регистр скорости передачи USART, старший байт
UCSRC		Регистр управления и состояния USART

В АЛУ выполняются арифметические и логические операции. Операнды поступают из регистров общего назначения. При выполнении одноместных операций результат записывается в регистр, из которого поступил операнд. При выполнении двухместных операций результат записывается в регистр, из которого поступил первый операнд.

1.4. Способы адресации и команды пересылки данных

Прямая адресация – это простейший вид адресации операнда в памяти, так как эффективный адрес содержится в самой коман-

де и для его формирования не используется никаких дополнительных источников или регистров. Главным элементом кода команды является код операции (КОП), что определяет, какие действия будут выполнены по данной команде. Допускается использование прямой адресации при обращении, как к основной, так и к регистровой памяти.

Непосредственная адресация – это когда в команде содержится не адрес операнда, а непосредственно сам операнд. Непосредственная адресация позволяет повысить скорость выполнения операции, так как в этом случае вся команда, включая операнд, считывается из памяти одновременно и на время выполнения команды хранится в процессоре в специальном регистре команд. Однако при использовании непосредственной адресации появляется зависимость кодов команд от данных, что требует изменения программы при каждом изменении непосредственного операнда.

При **косвенной адресации** адресная часть команды указывает адрес ячейки памяти или номер регистра, в которых содержится адрес операнда. Применение косвенной адресации операнда из оперативной памяти при хранении его адреса в регистровой памяти существенно сокращает длину поля адреса, одновременно сохраняя возможность использовать для указания физического адреса полную разрядность регистра.

Недостаток этого способа в том, что необходимо дополнительное время для чтения адреса операнда. Вместе с тем он существенно повышает гибкость программирования. Изменяя содержимое ячейки памяти или регистра, через которые осуществляется адресация, можно, не меняя команды в программе, обрабатывать операнды, хранящиеся по разным адресам.

Косвенная адресация не применяется по отношению к операндам, находящимся в регистровой памяти.

Относительная адресация используется тогда, когда память логически разбивается на блоки, называемые сегментами. В этом случае адрес ячейки памяти содержит две составляющих: адрес начала сегмента (базовый адрес) и смещение адреса операнда в сегменте. Адрес операнда определяется как сумма базового адреса и смещения относительно этой базы.

Для задания базового адреса и смещения могут применяться ранее рассмотренные способы адресации. Как правило, базовый адрес находится в одном из регистров регистровой памяти, а смещение может быть задано в самой команде или регистре. Адресное поле команды состоит из двух частей, в одной указывается номер регистра, хранящего базовое значение адреса (начальный адрес сегмента), а в другом адресном поле задается смещение, определяющее положение ячейки относительно начала сегмента. Именно такой способ представления адреса обычно и называют относительной адресацией.

Относительная адресация с индексированием где первая часть адресного поля команды также определяет номер базового регистра, а вторая содержит номер регистра, в котором находится смещение. Такой способ адресации чаще всего называют базово-индексным.

Главный недостаток относительной адресации это большое время вычисления физического адреса операнда. Но существенное преимущество этого способа заключается в возможности создания "перемещаемых" программ, которые можно размещать в различных частях памяти без изменения команд программы. То же относится к программам, обрабатывающим по единому алгоритму информацию, расположенную в различных областях запоминающего устройства. В этих случаях достаточно изменить содержимое базового адреса начала команд программы или массива данных, а не модифицировать сами команды.

Команды этой группы предназначены для пересылки содержимого ячеек, находящихся в адресном пространстве памяти данных. Разделение адресного пространства на три части (РОН, РВВ, память данных) 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.

Кроме регистров, МК может иметь память данных, обращение к которой производится при помощи регистровых пар (индексная адресация) или указанием 16-ти разрядного адреса. МК может только прочесть память данных в регистр или записать туда из регистра, никакие арифметические или логические операции с памятью данных невозможны. Обозначение операндов команд представлены в таблице 3.

Таблица 3

Обозначение операндов команд

Rd	регистр - приемник, место, куда сохраняется результат выполнения команды
Rs	регистр - источник в двухоперандных командах. Его значение после выполнения команды не изменяется.
I/O	регистр ввода-вывода, или периферия. Это порты, таймеры и т.д.
K	8-ми разрядная константа в операциях со "старшими" регистрами общего назначения (R16-R31)
b	Номер бита в операциях с регистрами ввода-вывода
A	16-ти разрядный адрес при работе с памятью данных
q	6-ти разрядное смещение при работе с памятью данных
X	Регистровая пара X. Состоит их регистров XL (R26) и XH (R27)
Y	Регистровая пара Y. Состоит их регистров YL (R28) и YH (R29)
Z	Регистровая пара Z. Состоит их регистров ZL (R30) и ZH (R31)

Команды пересылки данных и примеры способов адресации для МК ATmega8515 представлены в таблице 4.

Таблица 4

Команды пересылки данных

Мнемоника	Описание	Операция
MOV Rd, Rs	Пересылка между РОН	Эта команда копирует содержимое регистра Rs в регистр Rd. Содержимое Rs не изменяется, предыдущее содержимое Rd теряется. **Пример:** **mov R3,R19** ;содержимое R19 копируется в R3 работает со всеми регистрами.
MOVW Rd, Rs	Пересылка 2-байтных значений	
LDI Rd, K	Загрузка константы в РОН	Загружает в регистр Rd 8-ми разрядную константу. Работает со старшими регистрами (R16-R31). **Пример: ldi R16,1** ;загружает в R16 значение 1. Если необходимо загрузить константу в младший регистр, то это делается двумя командами: **ldi R16,1** ; загружает в R16 значение 1 **mov R4,R16** ;и копирует в R4
LD Rd, X	Косвенное чтение	Загружает в регистр Rd байт из памяти данных, адрес ячейки памяти в регистровой паре X. Содержимое регистровой пары X не изменяется. **Например: ldi XL,0** ;загружает младший байт регистровой пары X **ldi XH,2** ;-//- старший байт регистровой пары X **ld R5,X** ;байт из ОЗУ с адресом 0x200 загружается в R5
LD Rd, X+	Косвенное чтение с постинкрементом	Аналогично предыдущей команде, но содержимое регистровой пары X после выполнения пересылки данных увеличивается на 1. **Например: ldi XL,0** ;загружает младший байт регистровой пары X **ldi XH,2** ;-//- старший байт регистровой пары X **ld R5,X+** ;байт из ОЗУ с адресом 0x200 загружается в R5 **ld R6,X+** ;байт из ОЗУ с адресом 0x201 загружается в R6
LD Rd, -X	Косвенное чтение с прединкрементом	Аналогично предыдущей команде, но содержимое регистровой пары X перед выполнением пересылки данных уменьшается на 1. **Например: ldi XL,0** ;загружает младший байт регистровой пары X **ldi XH,2** ;-//- старший байт регистровой пары X **ld R5,-X** ;байт из ОЗУ с адресом 0x1FF загружается в R5 **ld R6,-X** ;байт из ОЗУ с адресом 0x1FE загружается в R6

Мнемоника	Описание	Операция
LD Rd, Y	Косвенное чтение	Эти команды работают абсолютно идентично трем ранее описанным, за исключением того, что индексным регистром является не X, а Y и Z. Наличие трех пар регистров дает возможность эффективной работы с блоками памяти, **например: ldi XL,0x00** ;первый блок памяти **ldi XH,0x02** ; регистровая пара X указывает на адрес 0x200 **ldi YL,0x80** ;второй блок памяти **ldi YH,0x01** ;регистровая пара Y указывает на адрес 0x180 **ldi R16,10** ;счетчик на 10 **LOOP**: **ld R5,X+** ;в R5 из первого блока, X указывает на следующий **st Y+,R5** ;из R5 во второй блок, Y также - на следующий **dec R16** ;и так - 10 раз **brne LOOP** В результате выполнения этого цикла 10 байт памяти, начиная с адреса 0x200 будут скопированы в область памяти с адресом 0x180
LD Rd, Y+	Косвенное чтение с постинкрементом	
LD Rd, -Y	Косвенное чтение с предекрементом	
LD Rd, Z	Косвенное чтение	
LD Rd, Z+	Косвенное чтение с постинкрементом	
LD Rd, -Z	Косвенное чтение с преддекрементом	
LDD Rd, Y+q	Косвенное относительное чтение	Регистровые пары Y и Z, кроме вышеописанных методов обращения к памяти данных, имеют еще один. В этом случае в регистр Rd загружается байт из ячейки памяти, чей адрес вычисляется как содержимое регистровой пары плюс 6-ти разрядное смещение. **Например: ldi YL,0** **ldi YH,2** ;регистровая пара Y указывает на адрес 0x200 **ldd R5,Y+5** ;байт из ОЗУ с адресом 0x205 загружается в R5 **ldd R6,Y+10** ;байт из ОЗУ с адресом 0x210 загружается в R6 Такой режим адресации невозможен для регистровой пары X. Значение смещения q - от 0 до 63.
LDD Rd, Z+q	Косвенное относительное чтение	
LDS Rd, k	Непосредственное чтение из ОЗУ	Команда LDS загрузит в регистр Rd содержимое ячейки памяти данных с адресом A, где A - шестнадцатиразрядная константа. В этом случае нет необходимости предварительно загружать регистровую пару, но сама команда займет два слова программной памяти. **Например:** **lds R5,0x240** ;байт из ОЗУ с адресом 0x240 загружается в R5
ST X, Rs	Косвенная запись	
ST X+, Rs	Косвенная запись с постинкрементом	
ST –X, Rs	Косвенная запись с преддекрементом	
ST Y, Rs	Косвенная запись	
ST Y+, Rs	Косвенная запись с постинкрементом	
ST –Y, Rs	Косвенная запись с преддекрементом	

Мнемоника	Описание	Операция
STD Y+q, Rs	Косвенная относительная запись	**sts 0x060,R5** ;байт R5 в ОЗУ с адресом 0x060 Парой для команды LDS является команда STS - записывающая содержимое регистра в память.
ST Z, Rs	Косвенная запись	
ST Z+, Rs	Косвенная запись с постинкрементом	
ST –Z, Rs	Косвенная запись с преддекрементом	
STD Z+q, Rs	Косвенная относительная запись	
STS k, Rs	Непосредственная запись в ОЗУ	
LPM	Загрузка данных из памяти программ-	
LPM Rd, Z	Загрузка данных из памяти программ	К командам пересылки данных надо отнести и очень специфичную команду LPM, которая пересылает в R0 байт памяти программ, на который указывает регистровая пара Z. Напомню, что память программ и память данных между собой никак не пересекаются. Данная команда используется в основном для чтения таблиц констант, располагаемых в памяти программ. **Например:** **TABLE: db 4,6,8,2,3,5,0** ;..... **ldi ZL,low(TABLE*2)** **ldi ZH,hi(TABLE*2)** **LPM** ;в R0 будет занесено число 4 Содержимое регистровой пары Z не изменяется, биты признаков - тоже. Вообще, ни одна команда пересылки данных не изменяет признаков.
LPM Rd, Z+	Загрузка данных из памяти программ с постинкрементом	
ELPM	Расширенная загрузка данных из памяти программ	
ELPM Rd, Z	Расширенная загрузка данных из памяти программ	
ELPM Rd, Z+	Расширенная загрузка данных из памяти программ с постинкрементом	
SPM	Запись в память программ	
IN Rd, A	Пересылка из PBB в POH	Команда IN прочтет байт из регистра ввода-вывода в регистр общего назначения, **например**: **in R18,PINA** ;прочитать состояние входных линий порта A в R18 **in R1,TCCR0** ;прочитать в R1 счетчик таймера 0
OUT A, Rs	Пересылка из POH в PBB	А эта - из регистра выведет в порт.
PUSH Rs	Сохранение байта в стеке	Эти команды предназначены для работы со стеком. Команда PUSH поместит Rs в стек, после выполнения команды указатель стека уменьшается на единицу. Команда POP извлечет байт из стека и поместит его в Rd. Соответственно, указатель стека увеличится на единицу.
POP Rd	Извлечение байта из стека	

1.5. Команды арифметических и логических операций

К данной группе относятся команды, позволяющие выполнять такие базовые операции, как сложение, вычитание, инкрементирование, декрементирование, а так же умножение. Все операции производятся только над регистрами общего назначения. При этом МК AVR позволяют легко оперировать как знаковыми, так и беззнаковыми числами, а так же 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.

При выполнении арифметических и логических операций происходит изменение значений определенных признаков в регистре SREG. Регистр SREG находится в области регистров ввода-вывода, по адресу 0x3F (0x5F). Чтение и запись производится командами IN/OUT, кроме того, есть специальные команды установки и очистки конкретного бита в SREG.

Принимающие значения регистра SREG имеются следующие биты представлено в таблице 5:

Таблица 5

Установка признаков при выполнении команд процессора ATmega8515

Признаки	Регистр	Операция
I	**SREG.7**	Бит разрешения прерывания. Если он = 0, то все прерывания в МК запрещены. Если он =1, то разрешением прерываний будут управлять соответствующие биты периферии.
T	**SREG.6**	Битовый аккумулятор. С этим битом работают команды BST и BLD
H	**SREG.5**	Флаг переноса из младшей тетрады
S	**SREG.4**	Sign - ислючающее ИЛИ битов N и V
V	**SREG.3**	Verflow - переполнение
N	**SREG.2**	Negative - Результат операции < 0
Z	**SREG.1**	Zero - Результат операции равен нулю
C	**SREG.0**	Carry - Флаг переноса

Команды арифметических и логических операций и их примеры для ATmega8515 представлены в таблице 6.

Таблица 6

Команды арифметических и логических операций для ATmega8515

Мнемоника	Операция
ADD Rd,Rs	Сложение Rd и Rs, результат помещается в Rd. Изменяемые признаки: H V N Z C
ADC Rd,Rs	То же, что и ADD, но еще прибавляется C-разряд. Используется при работе с числами разрядностью более байта. Например: **add R18,R20** ;сложили мл байты - может быть перенос **adc R19,R21** ;сложили старшие с учетом этого переноса Изменяемые признаки: H V N Z C
ADIW Rdl,q	Сложение пары регистров с константой (q - от 0 до 63). Работает с четырьмя старшими парами регистров, то есть Z,Y,X и R25:R24 и используется в основном для операций с указателями. Изменяемые признаки: V N Z C
SUB Rd,Rs	Вычитание Rs из Rd, результат помещается в Rd. Изменяемые признаки: H V N Z C
SUBI Rd,K	Вычитание из Rd константы K. Изменяемые признаки: H V N Z C. Если нужно прибавить к регистру, например, число 10 - следует написать **subi R16, -10**
SBC Rd,Rs	Вычитание Rs из Rd с учетом переноса. Результат в Rd. Изменяемые признаки: H V N Z C
SBCI Rd,K	Вычитание константы K из Rd с учетом переноса. Результат в Rd. Работает со старшими регистрами. Изменяемые признаки: H V N Z C
SBIW Rdl,q	Вычитание из пары регистров константы. См. описание ADIW
AND Rd,Rs	Логическое "И" Rd и Rs, результат помещается в Rd. Изменяемые признаки: V N Z Суть логического "И" - в Rd будут установлены в состояние лог. 1 те биты, которые были равны 1 и в Rd и в Rs, остальные сбрасываются в 0
ANDI Rd,K	То же, только вместо Rs - константа K. Работает со старшими регистрами
OR Rd,Rs	Логическое "ИЛИ" Rd и Rs, результат помещается в Rd. Изменяемые признаки: V N Z Суть логического "ИЛИ" - в Rd будут установлены в состояние лог. 1 те биты, которые были равны 1 или в Rd, или в Rs, остальные сбрасываются в 0
ORI Rd,K	Логическое "ИЛИ" Rd и константы K. Работает со старшими регистрами
EOR Rd,Rs	Исключающее "ИЛИ" Rd и Rs, результат помещается в Rd. Изменяемые признаки: V N Z Суть исключающего "ИЛИ" - в Rd будут установлены в состояние лог. 1 те биты, которые были не равны в Rd, и в Rs, Следует заметить, что нет команды "исключающее ИЛИ" с константой!

Мнемоника	Операция
COM Rd	Изменит все биты Rd на противоположные. Изменяемые признаки: V N Z C
NEG Rd	Изменение знака Rd. Вычисляется как 0x00 - Rd Изменяемые признаки: H V N Z C
INC Rd **DEC** Rd	Инкремент / декремент Rd. Думаю, тут все ясно... Изменяемые признаки: N Z V
TST Rs	Установка признаков по содержимому Rs. На самом деле вычисляется как AND Rs, Rs. Изменяемые признаки: N Z V
CLR Rd	Очистка Rd (занесение в Rd нуля). Выполняется как EOR Rd,Rd , поэтому изменяет признаки: N Z V
SER Rd	Занесение константы 0xFF в Rd. Именно так и выполняется - LDI Rd, 0xFF

Команды сдвигов, установок разрядов портов и регистра состояния процессора ATmega8515 и их примеры представлены в таблице 7.

Таблица 7

Команды сдвигов для ATmega8515

Мнемоника	Операция
LSL Rd	Логический сдвиг содержимого регистра влево. Старший бит выдвигается в C разряд SREG, на его место становится 6-й бит, на место 6-го - 5-й и так далее. В самый младший - задвигается 0 До выполнения: C B7 B6 B5 B4 B3 B2 B1 B0 После выполнения: B7 B6 B5 B4 B3 B2 B1 B0 0 Изменяет признаки: Z,C,N,V,H
LSR Rd	До выполнения: B7 B6 B5 B4 B3 B2 B1 B0 C После: 0 B7 B6 B5 B4 B3 B2 B1 B0 Изменяет признаки: **Z,C,N,V**
ROL Rd	Циклический сдвиг содержимого регистра влево. Отличается от LSL тем, что в нулевой бит задвигается C-разряд: До выполнения: C B7 B6 B5 B4 B3 B2 B1 B0 После выполнения: B7 B6 B5 B4 B3 B2 B1 B0 C Изменяет признаки: **Z,C,N,V,H**

Мнемоника	Операция
ROR Rd	До выполнения: B7 B6 B5 B4 B3 B2 B1 B0 C После: C B7 B6 B5 B4 B3 B2 B1 B0 Изменяет признаки: **Z,C,N,V**
ASR Rd	Арифметический сдвиг вправо - иными словами, целочисленное деление на 2. Старший бит повторяет сам себя - поскольку это знак. До выполнения: B7 B6 B5 B4 B3 B2 B1 B0 C После: B7 B7 B6 B5 B4 B3 B2 B1 B0 Изменяет признаки: **Z,C,N,V**
SWAP Rd	Обмен тетрад До выполнения: B7 B6 B5 B4 B3 B2 B1 B0 После: B3 B2 B1 B0 B7 B6 B5 B4 Признаки не изменяются
SBI IO,b	Установить в "1" бит с номером b (b=0..7) в регистре ввода-вывода IO. Признаки не изменяются.
CBI IO,b	То же самое - только установить в "0"
BST Rs,b	Скопирует бит b регистра **Rs** в бит T **SREG** (регистр состояния)
BLD Rd,b	Бит T SREG занесет в бит b регистра Rd. Эти две команды позволяют переставлять биты как угодно, жаль только, что нет команды инверсии T-бита
Далее следуют 16 команд установки или сброса битов признаков SREG. **Например** SEC - Set C - установить признак C в единицу, CLC - Clear C - в ноль.	
SEC CLC SEN CLN SEZ CLZ SEI CLI **SES CLS SEV CLV SET CLT SEH CLH**	
NOP	Пустая операция. Не делает ничего, кроме того, что занимает один такт процессора. Имеет код операции 0x000, что дает возможность "забить" ею любую другую команду без стирания всей программы (подробнее об этом чуть позже)
SLEEP	Перевод процессора в режим пониженного энергопотребления.
WDR	Сброс сторожевого таймера.

Программа для любого МК представляет собой последовательность команд, записанных в памяти программ. Большинство

команд при выполнении изменяют содержимое одного или нескольких РОН, РВВ или ячеек памяти данных.

Для обращения к различным областям адресного пространства памяти данных используются различные команды, реализующие, в свою очередь, различные способы адресации.

Далее представлен пример программы на языке ассемблера для процессоров ATmega8515. Демонстрация использования светодиодов и кнопок в составе STK500. При нажатии кнопки sw0 загорается светодиод LED0, кнопки sw1 загорается светодиод LED1, кнопки sw2 светодиоды гаснут.

```
.include "8515def.inc"
.def Temp =r16 ; Регистр хранения временных данных
.def Delay =r17 ; Переменная 1 для генерации задержки
.def Delay2 =r18 ; Переменная 2 для генерации задержки
;***** Инициализация
RESET:
ser temp
out DDRC, temp
;**** Тестирование ввода/вывода
LOOP:
sbis PIND,0x00 ; Если PortD.0 = 0,
sbi PORTC,0x00
sbis PIND,0x01 ; Если PortD.1 = 0,
sbi PORTC,0x01
sbis PIND,0x02 ; Если PortD.2 = 0
rjmp m1

;**** Далее необходима задержка
DLY4:
dec Delay
brne DLY4
dec Delay2
```

```
brne DLY4
rjmp LOOP ; Повторение цикла заново

m1:
cbi PORTC,0x00
cbi PORTC,0x01
rjmp DLY4
```

2. Порядок выполнения работы

2.1. Изучить описание структуры и функционирования процессора ATmega8515.

2.2. Ознакомиться с режимами основного меню интегрированный отладчик AVR Studio, набором операций, выполняемых в каждом из режимов.

2.3. Подготовить лабораторный макет к работе, выполнив следующие операции:

- включить источник питания платы и компьютер;
- ознакомиться с руководством пользователя стартового набора STK500;
- запустить из рабочей папки интегрированный отладчик AVR Studio;
- создать новый проект и написать приведенный выше пример программы для процессоров ATmega8515;
- загрузить рабочую программу в процессор ATmega8515, в соответствии с руководством пользователя в стартовый набора STK500, проверить работоспособность.

2.4. После загрузки и выполнения рабочей программы, выйти из программы AVR Studio и выключить питание лабораторного макета.

2.5. Разобраться с примером программы на языке ассемблера для процессоров семейства ATmega8515, занести в отчет резуль-

тат выполненной работы с интегрированной среде программирования AVR Studio и пример программы с комментариями.

3. Содержание отчета

3.1. Наименование лабораторной работы.

3.2. Цель работы.

3.3. Теоретическую часть.

3.4. Текст выполненных программ.

3.5. Вывод.

Контрольные вопросы

1. Архитектурные особенности процессоров AVR семейства Mega.

2. Приведите общую структуру, состав и технические параметры МК ATmega8515.

3. Приведите регистровую модель и установку признаков при выполнении арифметических и логических команд процессора ATmega8515.

ЛАБОРАТОРНАЯ РАБОТА №2

Команды управления программой 8-разрядных процессоров ATmega

Цель работы:

Знакомство с реализуемыми процессором способами адресации и командами пересылки данных.

Знакомство с арифметическими и логическими командами, а так же операциями сравнения.

Знакомство с командами управления программой, выполняемых процессором ATmega8515.

Практическое освоение интегрированной среды программирования, изучение команд управления программой.

Используемое оборудование:

- персональная ЭВМ, совместимая с IBM PC.

Используемое программное обеспечение:

- операционная система Windows XP;
- интегрированная среда программирования AVR Studio4.

1. Краткое описание работы

1.1. Команды передачи управления программой

В эту группу входят команды перехода, вызова подпрограмм и возврата их них и команды типа «проверка/пропуск», пропускающие следующую за ними команду при выполнении некоторого условия. Также к этой группе относятся команды сравнения, формирующие признаки регистра SREG и предназначенные, как правило, для работы совместно с командами условного перехода.

В системе команд МК семейства имеются команды как безусловного, так и условного перехода. Команды относительного (RJMP), косвенного (IJMP, EIJMP) и абсолютного (JMP) безусловного перехода являются самыми простыми в этой группе. Их функция заключается только в записи нового адреса в счетчик команд, однако это изменение происходит только при выполнении некоторого условия или, точнее, при определенном состоянии различных признаков регистра SREG (табл.8).

Таблица 8

Команды передачи управления программой

Мнемоника	**Описание**	**Операция**	**Признаки**
RJMP k	Относительный безусловный	PC = PC + k + 1	-
IJMP	Косвенный безусловный переход	PC = Z	-
EIJMP	Расширенный косвенный безусловный переход	PC = EIND:Z	-
JMP k	Абсолютный переход	PC = k	-
RCALL k	Относительный вызов подпрограммы	PC = PC + k + 1	-
ICALL	Косвенный вызов подпрограммы	PC = Z	-
EICALL	Расширенный косвенный вызов подпрограммы	PC = EIND.Z	-
CALL k	Абсолютный вызов подпрограммы	PC = k	-
RET	Возврат из подпрограммы	PC = STACK	-
RETI	Возврат из подпрограммы 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

Мнемоника	Описание	Операция	Признаки
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 A,b	Пропуск следующей команды, если бит РВВ сброшен	Если A.b = 0, то PC = PC + 2 (3)	-
SBIS A,b	Пропуск следующей команды, если бит РВВ установлен	Если A.b = 1, то PC = PC + 2(3)	-
BRBC s,k	Переход, если флаг s регистра SREG сброшен	Если SREG.s = 0, то PC = PC + k + 1	-
BRBS s,k	Переход, если флаг s регистра SREG установлен	Если SREG.s=1, то PC = PC + к + 1	-
BRCS k	Переход по переносу	Если C = 1, то PC = PC + к+1	-
BRCC k	Переход, если нет переноса	Если C = 0, то PC = PC + к+1	-
BREQ k	Переход по «равно»	Если Z= 1, то PC = PC + к + 1	-
BRNE k	Переход по «не равно»	Если Z = 0, то PC = PC + к + 1	-
BRSH k	Переход по «больше или равно»	Если C = 0, то PC = PC + к+1	-
BRLO k	Переход по «меньше»	Если C = 1, то PC = PC + к + 1	-
BRMI	Переход по «отрицательное значение»	Если N = 1, то PC = PC + к + 1	-
BRPL	Переход по «положительное значение»	Если N = 0, то PC = PC + k + 1	-

Мнемоника	Описание	Операция	Признаки
BRGE	Переход по «больше или равно» (числа со знаком)	Если (N + V) = 0, то PC = PC + k + 1	-
BRLT	Переход по «меньше нуля» (числа со знаком)	Если (N+V)=l, то PC = PC + k + 1	-
BRHS	Переход по половинному переносу	Если H = 1, то PC = PC + k + 1	-
BRHC	Переход, если нет половинного переноса	Если H = 0, то PC = PC + k + 1	-
BRTS	Переход, если флаг T установлен	Если T= 1, то PC = PC + k + 1	-
BRTC	Переход, если флаг T сброшен	Если T = 0, то PC = PC + k + 1	-
BRVS	Переход по переполнению дополнительного кода	Если V=1, то PC = PC + k + 1	-
BRVC	Переход, если нет переполнения дополнительного кода	Если V = 0, то PC = PC + k + 1	-
BRID	Переход, если прерывания запрещены	Если I = 0, то PC = PC + k + 1	-
BRIE	Переход, если прерывания разрешены	Если I = 1, то PC = PC + k + 1	-

1.2. Прерывания, выполняемые процессором ATmega

Прерывание (англ. interrupt) – сигнал, сообщающий процессору о наступлении какого-либо события. При этом выполнение текущей последовательности команд приостанавливается, и управление передается обработчику прерывания, который выполняет работу по обработке события и возвращает управление в прерванный код.

Прерывание прекращает нормальный ход программы для выполнения приоритетной задачи, определяемой внутренним или

внешним событием МК. При возникновении прерывания МК сохраняет в стеке содержимое счетчика команд PC и загружает в него адрес соответствующего вектора прерывания. По этому адресу, как правило, находится команда безусловного перехода RJMP к подпрограмме обработки прерывания.

Последней командой подпрограммы обработки прерывания должна быть команда RETI, которая осуществляет возврат в основную программу и восстановление предварительно сохраненного счетчика команд.

МК AVR семейства Mega имеют многоуровневую систему приоритетных прерываний. Младшие 20 адресов памяти программ, начиная с адреса 0x0001, отведены под таблицу векторов прерываний. Каждому прерыванию соответствует адрес в этой таблице, который загружается в счетчик команд при возникновении прерывания. Положение вектора в таблице также определяет и приоритет соответствующего прерывания: чем меньше адрес, тем выше приоритет прерывания. Размер вектора прерывания зависит от объема памяти программ МК и составляет 1 байт.

Положение таблицы векторов прерываний может быть изменено. Таблица может располагаться не только в начале памяти программ, но и в начале области загрузчика, причем перемещение таблицы может быть осуществлено непосредственно в ходе выполнения программы. Для управления размещением таблицы прерываний используется регистр управления МК GICR.

INT1 – разрешение внешнего прерывания INT1: если в этом бите записана логическая «1» и флаг I регистра SREG также установлен в «1», то разрешается внешнее прерывание с вывода INT1.

INT0 – разрешение внешнего прерывания INT0: если в этом бите записана логическая «1» и флаг I регистра SREG также ус-

тановлен в «1», то разрешается внешнее прерывание с вывода INT0.

INT2 – разрешение внешнего прерывания INT2: если в этом бите записана «1» и флаг I регистра SREG установлен в «1», то разрешается внешнее прерывание с вывода INT2.

IVSEL – определяет положение таблицы векторов прерываний в памяти программ. Если флаг сброшен, то таблица векторов прерываний располагается в начале памяти программ, если установлен в «1» – в начале области загрузчика. Конкретное значение начального адреса области загрузчика зависит от установок конфигурационной ячейки BOOTRST.

IVCE – предназначен для разрешения изменения флага IVSEL.

Для изменения положения таблицы векторов прерываний необходимо выполнить следующие действия:

1. Установить бит IVCE в «1».

2. В течение следующих четырех тактов занести требуемое значение в бит IVSEL, при этом бит IVCE сбрасывается в «0». В противном случае бит IVCE будет сброшен аппаратно по истечении четырех тактов, запрещая дальнейшее изменение флага IVSEL.

На время выполнения описанной последовательности прерывания автоматически запрещаются, и разрешаются только после сброса флага IVCE.

Состояние флага I регистра SREG при этом не меняется. В таблице 9 приведена таблица прерываний МК ATmega8535.

Таблица 9

Таблица векторов прерываний микроконтроллера ATmega8535

№	Адрес	Источник	Описание
1	0x0000	RESET	Внешний сброс, сброс при включении питания, сброс монитора напряжения питания, сброс сторожевого таймера
2	0x0001	INT0	Внешнее прерывание 0
3	0x0002	INT1	Внешнее прерывание 1

№	Адрес	Источник	Описание
4	0x0003	TIMER2_COMP	Совпадение таймера/счетчика T2
5	0x0004	TIMER2_OVF	Переполнение таймера/счетчика T2
6	0x0005	TIMER1_CAPT	Захват таймера/счетчика T1
7	0x0006	TIMER1_COMPA	Совпадение A таймера/счетчика T1
8	0x0007	TIMER1_COMPB	Совпадение B таймера/счетчика T1
9	0x0008	TIMER1_OVF	Переполнение таймера/счетчика T1
10	0x0009	TIMER0_OVF	Переполнение таймера/счетчика T0
11	0x000A	SPI, STC	Передача по SPI завершена
12	0x000B	USART, RXC	USART, прием завершен
13	0x000C	USART, UDRE	Регистр данных USART пуст
14	0x000D	USART, TXC	USART, передача завершена
15	0x000E	ADC	Преобразование АЦП завершено
16	0x000F	EE_RDY	EEPROM готово
17	0x0010	ANA_COMP	Аналоговый компаратор
18	0x0011	TW1	Прерывание от модуля TWI
19	0x0012	INT2	Внешнее прерывание 2
20	0x0013	TIMER0_COMP	Совпадение таймера/счетчика T0
21	0x0014	SPM_RDY	Готовность SPM

Для разрешения прерываний должен быть установлен флаг I регистра SREG.

Индивидуальное разрешение или запрет прерываний осуществляется установкой или сбросом соответствующих битов регистров масок. При возникновении прерывания флаг I регистра SREG аппаратно сбрасывается, запрещая тем самым обработку следующих прерываний. В программе обработчика прерывания можно снова установить этот флаг для разрешения вложенных прерываний. При возврате из подпрограммы обработки прерывания (при выполнении команды RETI) флаг I устанавливается аппаратно.

При вызове подпрограмм обработки прерываний регистр состояния SREG не сохраняется. Поэтому пользователь должен самостоятельно запоминать содержимое этого регистра при входе в подпрограмму обработки прерывания и восстанавливать его значение перед вызовом команды RETI.

Очередь прерываний работает следующим образом: если условия генерации одного или более прерываний возникают в то

время, когда флаг общего разрешения прерываний сброшен (все прерывания запрещены), соответствующие флаги устанавливаются в «1» и остаются в этом состоянии до установки флага общего разрешения прерываний. После разрешения прерываний выполняется их обработка в порядке приоритета.

Наименьшее время отклика для любого прерывания составляет 4 такта. В течение этого времени происходит сохранение счетчика команд в стеке. В течение последующих двух тактов выполняется команда перехода к подпрограмме обработки прерывания. Если прерывание произойдет во время выполнения команды, длящейся несколько циклов, то генерация прерывания произойдет только после выполнения этой команды. Если прерывание произойдет во время нахождения МК в «спящем» режиме, то время отклика увеличивается еще на 4 или 5 тактов. Возврат в основную программу занимает 4 такта, в течение которых происходит восстановление счетчика команд из стека. После выхода из прерывания процессор всегда выполняет одну команду основной программы, прежде чем обслужить любое отложенное прерывание.

Внешние прерывания генерируются при изменении состояния на выводах INT0, INT1, INT2 даже в случае, если указанные выводы сконфигурированы как выходы. Для маскирования внешних прерываний используются регистры MCUCR и MCUCSR.

SM2:SM0 – управление спящим режимом МК:

000 – режим холостого хода;

001 – режим снижения шумов АЦП;

010 – режим микропотребления;

011 – экономичный режим;

100 – зарезервировано;

101 – зарезервировано;

110 – режим ожидания;

111 – расширенный режим ожидания.

SE – разрешение перехода в режим пониженного энергопотребления: установка этого бита в «1» разрешает перевод МК в режим пониженного энергопотребления по команде SLEEP, при сброшенном бите SE выполнение команды SLEEP не производит никаких действий.

ISC11:ISC10, ISC01:ISC00 – определяют условия генерации внешних прерываний на выводах INT1/INT0:

00 – прерывание при низком уровне на выводе INT1/INT0;

01 – прерывание при любом изменении сигнала на выводе INT1/INT0;

10 – прерывание по спадающему фронту на выводе INT1/INT0;

11 – прерывание по нарастающему фронту на выводе INT1/INT0.

ISC2 – определяет условия генерации внешних прерываний на выводе INT2:

0 – прерывание по спадающему фронту на выводе INT2;

1 – прерывание по нарастающему фронту на выводе INT2.

WDRF –флаг сброса сторожевого таймера: устанавливается в «1» при сбросе сторожевого таймера, сбрасывается при включении питания или программно записью «0».

BORF – флаг сброса монитора питания: устанавливается в «1» при сбросе монитора питания, сбрасывается при включении питания или программно записью «0».

EXTRF – флаг внешнего сброса: устанавливается в «1» при внешнем сбросе, сбрасывается при включении питания или программно записью «0».

PORF – флаг сброса по питанию: устанавливается в «1» при включении питания, сбрасывается только программно записью «0».

2. Порядок выполнения работы

2.1. Изучить набор команд пересылки данных, выполняемых процессорами ATmega8515.

Изучить набор арифметических и логических команд, а так же операции сравнения, выполняемых процессорами ATmega8515.

Изучить набор команд управления программой, выполняемых процессорами ATmega8515.

2.2. Подготовить лабораторный макет к работе, выполнив следующие операции:

- включить источник питания платы и инструментальный компьютер;

- запустить из рабочего каталога интегрированный отладчик AVR Studio.

2.3. Выполнить набор команд безусловных переходов и обращения к подпрограмме в соответствии с таблицами 4-9. В качестве подпрограммы использовать последовательность команд пересылки и обработки данных (3 – 4 команды). Результат выполнения контролировать в окне Processor в пошаговом режиме Step into (F11) или Auto Step в меню Debug, (рис.4).

2.4. Выполнить набор команд условных переходов и реализации программных циклов, предварительно обеспечив выполнение соответствующих условий с помощью команд сравнения или обработки данных. Результат выполнения контролировать в окне Processor в пошаговом режиме Step into (F11) или Auto Step в меню Debug, (рис.4).

2.5. В процессе выполнения самостоятельно разработать и отладить программу на языке Ассемблера, реализующую (по заданию преподавателя) достаточно сложную процедуру обработки данных (до 40 – 50 операторов). Провести трассировку программы, продемонстрировать полученный результат.

2.6. После загрузки и выполнения рабочей программы, выйти из программы AVR Studio и выключить питание лабораторного макета.

2.7. Результат выполненной работы в интегрированной среде программирования занести в отчет.

3. Содержание отчета

3.1. Наименование лабораторной работы.

3.2. Цель работы.

3.3. Теоретическую часть.

3.4. Текст выполненных программ.

3.5. Вывод.

Контрольные вопросы

1. Способы адресации и команды ассемблера, выполняемые процессором семейства ATmega.

2. Приведите команды управления программой и установку признаков при выполнении процессоров семейства ATmega.

3. Приведите механизм выполнения подпрограммы и обработчика прерывания. Опишите команды ветвления к подпрограмме процессоров семейства ATmega приведите примеры.

ЛАБОРАТОРНАЯ РАБОТА №3

РЕАЛИЗАЦИЯ СИСТЕМЫ УПРАВЛЕНИЯ НА 8-РАЗРЯДНЫХ МИКРОКОНТРОЛЛЕРОВ ATMEGA8515

Цель работы:

Изучение цифровой системы управления микроконтроллеров семейства ATmega8515:

- вывод данных на светодиодные индикаторы;
- ввод данных с клавиатуры;
- вывод данных на жидкокристаллический индикатор.

Используемое оборудование:

- персональная ЭВМ, совместимая с IBM PC.

Используемое программное обеспечение:

- операционная система Windows XP;
- интегрированная среда программирования AVR Studio4.

1. Краткое описание работы

1.1. Вывод данных на светодиодные индикаторы

В набор STK500 входят 8 желтых светодиодов. Светодиоды электрически отделены от остальной части платы за счет подключения к собственным разъемам (см. приложение эл. принц. схема STK500). Таким образом, светодиоды подключены к AVR МК через 10-проводной шнур и разъем порт ввода/вывода (Port B). На рисунке 8 показано как реализовано управление светодиодом. Данное решение позволяет получить одинаковую интенсивность свечения светодиода при нахождении напряжения питания МК в диапазоне 1.8 В…5.0 В.

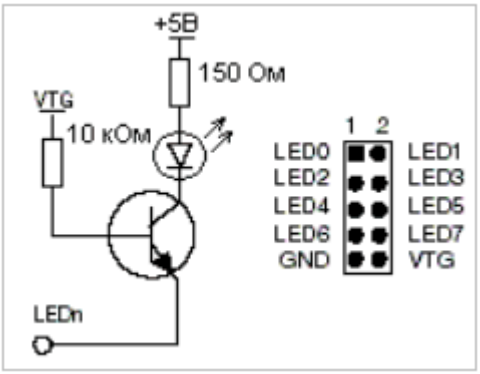

Рис. 8. Схема включения светодиода и подключение светодиодов к разъему

Порты AVR МК могут управлять непосредственно светодиодной нагрузкой, как втекающим током, так и вытекающим. Однако, в STK500 используются транзистор и два резистора для поддержания постоянной яркости свечения светодиодов при любом значении напряжения питания МК (VTG), а также при выключении светодиодов, когда VTG отсутствует. Программа управления светодиодами на языке ассемблера см. пример.

1.2. Ввод данных с клавиатуры

Кнопки электрически отделены от остальной части платы за счет подключения к собственным разъемам (см. приложение эл. принц. схема STK500). В наборе STK500 порт ввода-вывода (Port D) подключен разъем SW, к которому подключены кнопки. При нажатии на кнопку на выводе SWn будет низкий уровень напряжения, а при отпускании – высокий (VTG). Рабочий диапазон напряжения VTG =1.8…5.0В.

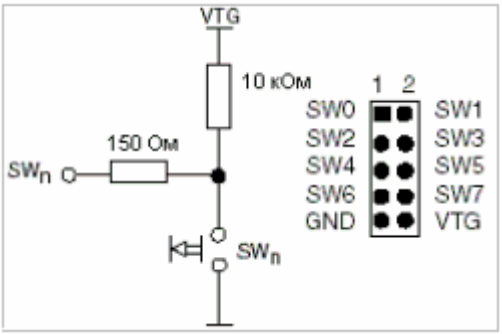

Рис. 9. Схема включения кнопок и подключения к разъему

На линиях портов ввода-вывода AVR МК имеется возможность активизации встроенных подтягивающих резисторов к плюсу питания. Это свойство можно использовать в целях исключения внешнего подтягивающего резистора. В STK500 добавлены внешние подтягивающие резисторы 10 кОм для формирования лог. «1» на выводах SWn при отжатом состоянии кнопок. Резистор 150 Ом выполняет функцию защитного токоограничения, например, в случае ошибочной настройки линий ввода-вывода, связанных с кнопками, на вывод.

1.3. Вывод данных на жидкокристаллический дисплей

В качестве устройства отображения информации на к STK500 подключен двухстрочный 40-символьный жидкокристаллический дисплей (ЖКД) фирмы WINSTAR типа WH-1602. На плате ЖКД смонтирован специализированный контроллер HD44780, который обеспечивает выдачу необходимых управляющих сигналов и генерацию на дисплее заданного набора символов (табл. 9). Сборка, состоящая из ЖКД и контроллера, называется ЖК-модулем. На рисунке 10 представлена структурная схема подключения ЖК-модуля к МК по 8-ми проводной системной шине.

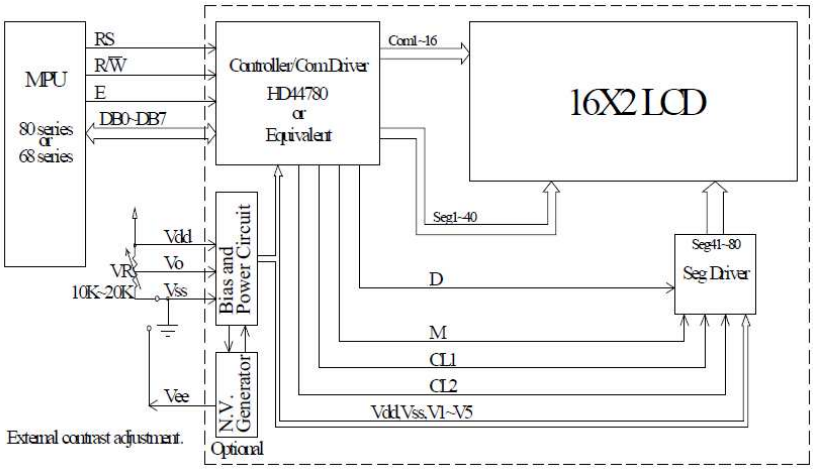

Рис. 10. Схема подключения ЖК-модуля к системной шине

Выводы DB7…DB0 это шина данных/адреса. E - стробирующий вход. RW – определяет в каком направлении движутся данные. Если 1 - то на чтение из ЖКД, если 0 то на запись в ЖКД. RS – определяет что у нас передается, команда (RS=0) или данные (RS=1). Данные будут записаны в память по текущему адресу, а команда исполнена МК.

Со стороны питания ЖК-модуль подключается следующим образом: GND – минус, он же общий; Vcc – плюс питания, обычно 5V; V0 – вход контрастности.

На плате имеется подстроечный резистор R, позволяющий менять значение напряжения Vo, что приводит к изменению ориентации жидких кристаллов на табло дисплея. Таким образом, можно регулировать контрастность изображения при определенном угле наблюдения (снизу-вверх или сверху-вниз).

Для вывода символа на дисплей МК должен по шине DB7-0 записать в регистр данных ЖК-модуля код соответствующего

символа (рис. 11). При этом код символа записывается в DDRAM по текущему адресу, заданному содержимым счетчика.

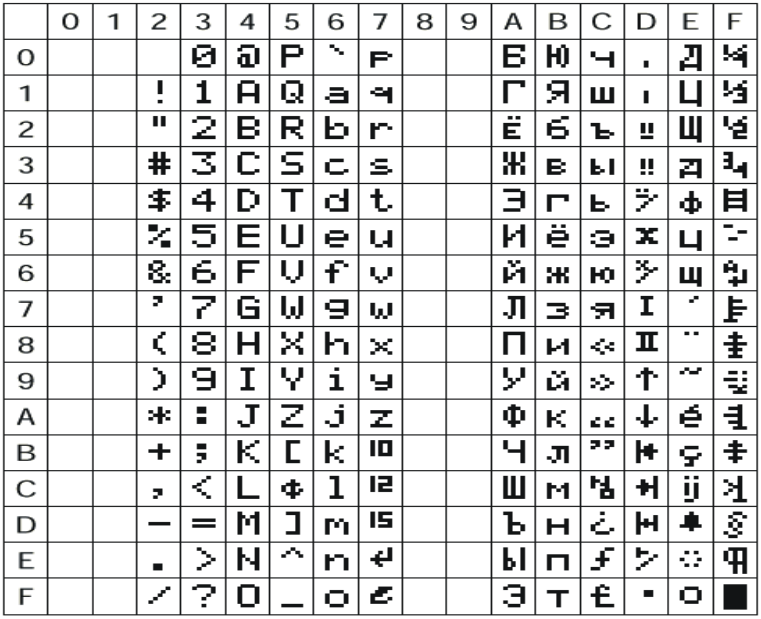

Рис. 11. Коды символов, представляемых на дисплее ЖКИ-модуля

Изображение каждого символа на дисплее реализуется с помощью матрицы 5x8 точек. Состояние каждой точки: включено (светло) или отключено (темно), определяется кодом соответствующего видеосимвола, хранящегося в памяти контроллера, входящего в состав ЖК-модуля. Этот контроллер содержит память видеосимволов DDRAM, в которой хранятся коды стандартных видеосимволов, изображаемых на дисплее, и ОЗУ знакогенератора CGRAM, в которое пользователь вводит коды дополнительных символов, представляемых на дисплее вместе со стандартными символами. Положение изображаемого на дисплее символа опре-

деляется адресным счетчиком AC, в котором содержится адрес позиции выводимого символа. Адреса символов 1-й строки AC = $00 - 28 (40 позиций), адреса символов второй строки $40 - 67 (40 позиций). При выводе символов на экран счётчик AC, дойдя до последнего символа 1-й строки (адрес $28), автоматически перейдёт на начало 2-й строки (адрес $40). Адресация AC к символам, имеющим адреса в диапазонах $28-3F и $68-7F запрещена (может привести к непредсказуемым результатам).

Для обращения к ЖК-модулю используется регистр команд IR, в который вводятся управляющие слов, и регистр данных DR, в который поступают данные для отображения на дисплее. Управление режимом работы ЖК-модуля производится путем установки значения управляющих флагов, которая выполняется путем ввода управляющих слов в регистр команд IR (табл. 9). Управляющие флаги имеют следующее назначение (в скобках указано их начальное значение, устанавливаемое при включении питания):

I/D (=1) - определяет изменение значения счетчика AC после вывода очередного символа: уменьшение на 1 при I/D=0, увеличение на 1 при I/D=1;

S (=0) - задает режим сдвига содержимого экрана: при S=0 сдвиг не производится, при S=1 изображение на экране после вывода очередного символа сдвигается вправо (при значении флага I/D=0) или влево (при значении флага I/D=1);

D/L (=1) - определяет разрядность используемой шины данных: 8-разрядная при значении D/L=1, 4-разрядная при D/L=0;

N (=0) - задает число используемых строк дисплея: 0 - одна строка, 1 - две строки;

F (=0) - указывает размер матрицы изображения символов: 0 - 5x8, 1 - 5x10 точек;

D (=0) - управляет выводом изображения на дисплей: 1 - вывод изображения, 1 - отсутствие изображения;

C (=0) - выводит на экран при C=1 изображение курсора в виде символа подчеркивания;

B (=0) - выводит на экран при B=1 представление курсора в виде мерцающего изображения символа (знакоместа).

Дополнительные флаги используются при выполнении сдвига изображения на дисплее с помощью соответствующего управляющего слова (табл. 8):

S/C - определяет объект смещения: при S/C=0 производится сдвиг курсора, при S/C=0 - сдвиг всего изображения;

R/L - задает направление сдвига курсора или изображения: влево при R/L=0, вправо при R/L=1.

Режим работы ЖК-модуля задается путем ввода в регистр команд IR управляющих слов, устанавливающих необходимые значения флагов. В табл. 10 приведены возможные значения управляющих слов и реализуемые ими функции. Два первых управляющих слова обеспечивают начальную установку счетчика адреса AC, четыре следующие слова производят установку необходимых значений флагов, два последних слова выполняют установку значения счетчика AC при обращении к внутренней памяти видеосимволов - CGRM или DDRAM.

Таблица 10.

Управляющие слова, задающие режим работы ЖКИ-модуля

D7 D6 D5 D4 D3 D2 D1 D0	Назначение
0 0 0 0 0 0 0 1	Очистка экрана, AC = 0, адресация AC на DDRAM
0 0 0 0 0 0 1 -	AC = 0, адресация на DDRAM, сброшены сдвиги, начало строки адресуется в начале DDRAM
0 0 0 0 0 1 I/D S	Выбирается направление сдвига курсора или экрана
0 0 0 0 1 D C B	Выбирается режим отображения
0 0 0 1 S/C R/L - -	Команда сдвига курсора/экрана
0 0 1 DL N F - -	Определение параметров развертки и ширины шины данных
0 1 AG AG AG AG AG AG	Присвоение счетчику AC адреса в области CGRAM
1 AD AD AD AD AD AD AD	Присвоение счетчику AC адреса в области DDRAM

При чтении содержимого регистра IR на шину данных DB7-0 выводится 8-разрядное слово состояния, старший бит которого содержит признак занятости BF, а в семи младших битах указывается текущее значение счётчика адреса AC. Признак занятости имеет значение BF=1, когда контроллер ЖК-модуля занят (производит вывод изображения на дисплей), и BF=0 – когда контроллер свободен (доступен для ввода новой информации). Так как процедура выдачи изображения на дисплей занимает достаточно длительное время, управляющий МК должен проверять значение бита BF перед каждым обращением к ЖК-модулю.

Далее представлен пример программы использования ЖКИ на языке ассемблера для микроконтроллеров ATmega8515.

```
;**** Управление ЖК-модулем с помощью микроконтроллеров AVR
;* Управление двухстрочным ЖК-табло(16 знаков в строке,
;* тип контроллера - HD44780) через ЖК-интерфейс модуля STK500.
;* Тактовая частота AVR: 4 МГц (стандарт STK500).
;* Подпрограммы:
;* InitLCD: Инициализация ЖК-интерфейса для 8-разрядных символов
;* SendCom: Передает команду на ЖК-модуль
;* SendDat: Передает символ на ЖК-модуль
;* OutText: Передает текстовую строку на ЖК-модуль
;* DefChar: Определяет пользовательский символ
;*****************************************************************
.nolist
.include "8515def.inc"
.list
;***** Регистровые переменные
.def  tmp1 = r16          ; Рабочий регистр 1
.def  tmp2 = r17          ; Рабочий регистр 2
.def  prm1 = r18          ; Передаваемый параметр
```

```
.def  tim1 = r19              ; Счетчик цикла 1
.def  tim2 = r20              ; Счетчик цикла 2
.def  Cnt  = r21; Вспомогательный счетчик

.equ  RS   = 6   ; Register Select = разряд 6 порта C (A14)
.equ  Ena  = 7   ; Enable Impuls = разряд 7 порта C (A15)
.equ  RW   = 6 ; R/W = разряд 6 порта D (/WR)
.equ  RD   = 7   ; /RD = разряд 7 порта D
.equ  BF   = 7   ; Флаг занятости = разряд 7 порта A

.cseg

rjmp  Initial     ; После сброса – к главной программе

InitLCD:
rcall wait5ms
rcall wait5ms
rcall wait5ms              ; Задержка на 15 мс после подачи питания
ldi   prm1,$30             ; $30: установка функции, 8 разрядов
rcall SendCom              ; Отправка команды на ЖК-модуль
rcall wait5ms              ; Задержка на 5 мс
ldi   prm1,$30             ; $30: установка функции, 8 разрядов
rcall SendCom              ; Повторяем команду
rcall wait5ms              ; Задержка на 5 мс
ldi   prm1,$30             ; $30: установка функции, 8 разрядов
rcall SendCom              ; Повторяем команду
rcall wait5ms              ; Задержка на 5 мс
rcall WaitBusy             ; Ожидаем готовности ЖК-модуля к приему
ldi   prm1,$38             ; $38: функция = 8 разрядов, 2 строки, 5x8
rcall SendCom              ; Отправляем команду
```

```
rcall wait150us          ; Задержка на 150 мкс
rcall WaitBusy           ; Ожидаем готовности ЖК-модуля к приему
ldi   prm1,$08           ; $08: отключение табло, курсора, мерцания
rcall SendCom            ; Передаем команду
rcall wait150us          ; Задержка на 150 мкс
rcall WaitBusy           ; Ожидаем готовности ЖК-модуля к приему
ldi   prm1,$01           ; $01: очистка табло
rcall SendCom            ; Передаем команду
rcall wait5ms            ; Задержка на 5 мс
rcall WaitBusy           ; Ожидаем готовности ЖК-модуля к приему
ldi   prm1,$06           ; $06: режим инкремента адреса, без сдвига
rcall SendCom            ; Передаем команду
rcall wait150us          ; Задержка на 150 мкс
ret

SendCom:                 ; Передача в ЖК-модуль команды
cbi   PortC,RS           ; Register Select = команда
rcall Ausgabe
ret

SendDat:                 ; Передача в ЖК-модуль байта данных
sbi   PortC,RS           ; Register Select = байт данных
rcall Ausgabe
ret

Ausgabe:                 ; Длина данных = 8 бит
out   PortA,prm1         ; Выдача команды/данных на порт A
sbi   PortC,Ena          ; Подаем импульс разрешения
nop
nop                      ; Длительность импульса – минимум 450 нс
```

```
cbi   PortC,Ena          ; Снимаем импульс разрешения
ret

OutText:                 ; Вывод символа <cnt> на табло
lsl   ZL
rol   ZH                 ; Указатель Z – на 1 позицию влево
OT1:
lpm                      ; Загружаем код символа в r0
mov   prm1,r0            ; Копируем код символа в prm1
rcall WaitBusy           ; Ожидаем готовности ЖК-модуля к приему
rcall SendDat            ; Передаем символ, автоинкремент адреса
adiw  ZL,1               ; Инкрементируем указатель Z
dec   cnt                ; Счетчик – 1
brne  OT1                ; Выводим все символы на ЖК-табло
ret

DefChar:                 ; Определение пользовательских символов
lsl   prm1               ; Умножаем prm1 на 2
lsl   prm1               ; Умножаем prm1 на 4
ldi   ZH,high(CharTab)   ; Адрес старшего байта шаблона символа
lsl   prm1               ; prm1 x 8
mov   ZL,prm1            ; Z -> адрес символа
rol   ZH                 ; Указатель Z – на 1 позицию влево
sbr   prm1,1<<6          ; Разряд 6 = 1 -> адрес в памяти CG
rcall WaitBusy           ; Ожидаем готовности ЖК-модуля к приему
rcall SendCom            ; Устанавливаем счетчик адреса памяти CG
ldi   cnt,8
DC1:
lpm                      ; Загружаем в r0 шаблон символа
mov   prm1,r0            ; Копируем шаблон символа в prm1
```

```
rcall WaitBusy          ; Ожидаем готовности ЖК-модуля к приему
rcall SendDat           ; Передаем шаблон, автоинкремент адреса
adiw  ZL,1              ; Инкремент указателя Z
dec   cnt               ; Счетчик – 1
brne  DC1               ; Копируем все 8 символов в память CG
ret

WaitBusy:               ; Ожидание готовности ЖК-модуля к приему
clr   tmp1
out   DDRA,tmp1         ; Порт A = вход
cbi   PortC,RS          ; Register Select = команда
sbi   PortD,RW          ; Направление = чтение из ЖК-модуля
WB1:
sbi   PortC,Ena         ; Подаем импульс разрешения
nop
nop                     ; После макс. 320 нс стабильности данных
in    tmp1,PinA         ; считываем BF и значение счетчика адреса
cbi   PortC,Ena         ; Снимаем импульс разрешения
nop
sbrc  tmp1,BF           ; Пропускаем следующую команду,
                        ;если ЖК-модуль готов (BF=0)
rjmp  WB1
cbi   PortD,RW          ; Направление записи = в ЖК-модуль
ser   tmp1
out   DDRA,tmp1         ; Порт A = выход
ret

Wait50us:               ; Такт системной синхронизации = 4 МГц
ldi   tim1,65           ; Загрузка счетчика
Wait51:
```

```
dec   tim1
brne  Wait51
ret

Wait150us:          ; Такт системной синхронизации = 4 МГц
rcall wait50us
rcall wait50us
rcall wait50us
ret

Wait5ms:            ; Такт системной синхронизации = 4 МГц
ldi   tim2,100      ; Загрузка счетчика
Wait501:
rcall wait50us
dec   tim2
brne  Wait501
ret

Wait38ms:           ; Такт системной синхронизации = 4 МГц
ldi   tim2,0        ; Загрузка счетчика
Wait381:
rcall wait150us
dec   tim2
brne  Wait381
ret

Wait5s:             ; Такт системной синхронизации = 4 МГц
ldi   cnt,130       ; Загрузка счетчика
Wait5s1:
rcall wait38ms
```

```
dec   cnt
brne  Wait5s1
ret

Initial:
ldi   tmp1,High(RamEnd)
out   sph,tmp1
ldi   tmp1,Low(RamEnd)
out   spl,tmp1          ; Инициализируем стек
ldi   tmp1,$3F          ; Разряды 7..6 = 0, остальные = 1
out   PortD,tmp1        ; /RD,/WR = 0, остальные разряды порта D=1
out   PortC,tmp1        ; Ena, RS = 0, остальные разряды порта C=1
ldi   tmp1,$C0          ; Разряд 7 = 1, разряд 6 = 1
out   DDRD,tmp1         ; Разряды 7..6 - выходы, остальные - входы
out   DDRC,tmp1         ; Разряды 7..6 - выходы, остальные - входы
ser   tmp1              ; Устанавливаем tmp1
out   DDRA,tmp1         ; Порт A - выход
rcall InitLCD           ; Инициализируем ЖК-модуль
clr   tmp2              ; Счетчик символов
Ini1:
mov   prm1,tmp2         ; ... переносим как код знака
rcall DefChar           ; Определяем символ для этого кода
inc   tmp2
cpi   tmp2,8     ; = 8?
brne  Ini1              ; Переход, если нет

Haupt:
ldi   prm1,$0C          ; Включаем табло, откл. курсор и мерцание
rcall WaitBusy          ; Ожидаем готовность ЖК-модуля к приему
rcall SendCom           ; Передаем команду
```

```
ldi   prm1,$01          ; Очищаем табло, курсор – в начало
rcall WaitBusy          ; Ожидаем готовность ЖК-модуля к приему
rcall SendCom           ; Передаем команду
ldi   ZH,high(Text3)
ldi   ZL,low(Text3)     ; Z указывает на начало текста
ldi   cnt,16            ; Текст состоит из 16 символов
rcall OutText           ; Выводим Text3

TestBlinken:
ldi   prm1,64           ; Адрес = начало второй строки
sbr   prm1,1<<7         ; Устанавливаем разряд 7 -> память DD
rcall WaitBusy          ; Ожидаем готовность ЖК-модуля к приему
rcall SendCom           ; Устанавливаем указатель адреса в DD-RAM
ldi   ZH,high(Text4)
ldi   ZL,low(Text4)     ; Z указывает на начало текста
ldi   cnt,16            ; Текст состоит из 16 символов
rcall OutText           ; Выводим Text4
ldi   prm1,64+5         ; Адрес 6-го символа во второй строке
sbr   prm1,1<<7         ; Устанавливаем разряд 7 -> память DD
rcall WaitBusy          ; Ожидаем готовность ЖК-модуля к приему
rcall SendCom           ; Устанавливаем указатель адреса в DD-RAM
rcall WaitBusy          ; Ожидаем готовность ЖК-модуля к приему
ldi   prm1,$0D          ; Включаем табло, мерцание, откл. курсор
rcall SendCom           ; Передаем команду
rcall Wait5s            ; Задержка на 5 секунд
ldi   prm1,$14          ; Смещаем курсор вправо
rcall WaitBusy          ; Ожидаем готовность ЖК-модуля к приему
rcall SendCom           ; Передаем команду
rcall Wait5s            ; Задержка на 5 секунд
ldi   prm1,$14          ; Смещаем курсор вправо
```

```
rcall WaitBusy          ; Ожидаем готовность ЖК-модуля к приему
rcall SendCom           ; Передаем команду
rcall Wait5s            ; Задержка на 5 секунд

TestCursor:
ldi   prm1,$0E          ; Табло и курсор – вкл., мерцание – откл.
rcall WaitBusy          ; Ожидаем готовность ЖК-модуля к приему
rcall SendCom           ; Передаем команду
ldi   prm1,64           ; Адрес = начало второй строки
sbr   prm1,1<<7         ; Устанавливаем разряд 7 -> память DD
rcall WaitBusy          ; Ожидаем готовность ЖК-модуля к приему
rcall SendCom           ; Устанавливаем указатель адреса в DDRAM
ldi   ZH,high(Text5)
ldi   ZL,low(Text5)     ; Z указывает на начало текста
ldi   cnt,16            ; Текст состоит из 16 символов
rcall OutText           ; Выводим Text5
ldi   prm1,64+1         ; Адрес 2-го символа во второй строке
sbr   prm1,1<<7         ; Устанавливаем разряд 7 -> память DD
rcall WaitBusy          ; Ожидаем готовность ЖК-модуля к приему
rcall SendCom           ; Устанавливаем указатель адреса в DDRAM
rcall Wait5s            ; Задержка на 5 секунд

Sonderzeichen:
rcall WaitBusy          ; Ожидаем готовность ЖК-модуля к приему
ldi   prm1,$01          ; Очищаем табло
rcall SendCom           ; Передаем команду
rcall WaitBusy          ; Ожидаем готовность ЖК-модуля к приему
ldi   prm1,$0C          ; Табло – вкл., курсор и мерцание – откл.
rcall SendCom           ; Передаем команду
ldi   ZH,high(Text1)
```

```
ldi   ZL,low(Text1)       ; Z указывает на начало текста
ldi   cnt,16              ; Текст состоит из 16 символов
rcall OutText             ; Выводим Text1
ldi   prm1,64             ; Адрес = начало второй строки
sbr   prm1,1<<7           ; Устанавливаем разряд 7 -> память DD
rcall WaitBusy            ; Ожидаем готовность ЖК-модуля к приему
rcall SendCom             ; Устанавливаем указатель адреса в DD-RAM
ldi   ZH,high(Text2)
ldi   ZL,low(Text2)       ; Z указывает на начало текста
ldi   cnt,16              ; Текст состоит из 16 символов
rcall OutText             ; Выводим Text2 (8 служебных знаков)
rcall Wait5s              ; Задержка на 5 секунд
rcall Wait5s              ; Задержка на 5 секунд
rjmp  Haupt               ; Бесконечный цикл

.org  $400
CharTab:
.db   $04,$04,$1F,$04,$04,$00,$1F,$00      ; Код символа $00: символ "+/-"
.db   $0C,$02,$04,$08,$0E,$00,$00,$00      ; Код символа $01: символ "^2"
.db   $0C,$02,$0C,$02,$0C,$00,$00,$00      ; Код символа $02: символ "^3"
.db   $18,$06,$01,$06,$18,$00,$1F,$00      ; Код символа $03: символ ">="
.db   $03,$0C,$10,$0C,$03,$00,$1F,$00      ; Код символа $04: символ "<="
.db   $04,$04,$1F,$04,$1F,$04,$04,$00      ; Код символа $05: символ "<>"
.db   $00,$1F,$00,$1F,$00,$1F,$00,$00      ; Код символа $06: символ "="
.db   $15,$15,$15,$15,$15,$15,$15,$00      ; Код символа $06: символ "  "
Text1:
.db   "Sonderzeichen:  "
Text2:
.db   0,' ',1,' ',2,' ',3,' ',4,' ',5,' ',6,' ',7,' '
Text3:
```

```
.db   "  Display-Test  "
Text4:
.db   "Char 6-8 blinken"
Text5:
.db   "Cursor-Pos. = 65"
```

2. Порядок выполнения работы

2.1. Изучить набор команд ввода данных с клавиатуры и вывод данных на ЖКИ, выполняемых микроконтроллерами семейства ATmega8515.

2.2. Подготовить лабораторный макет к работе, выполнив следующие операции:

- включить источник питания платы и инструментальный компьютер;
- запустить из рабочего каталога интегрированный отладчик AVR Studio.

2.3. Выполнить запись байта в регистр светодиодного индикатора и проверить правильность записи по состоянию светоиндикаторов.

2.4. Составить программу опроса клавиатуры, установив для определенных клавиш заданные преподавателем запись битов в регистр светодиодного индикатора.

2.5. Провести инициализацию ЖК-модуля, установив необходимый режим его работы.

2.6. Ввести в память контроллерной платы набор символов, заданных преподавателем, и обеспечить их отображение на ЖКИ.

2.7. Результат выполненной работы в интегрированной среде программирования занести в отчет.

3. Содержание отчета

3.1. Наименование лабораторной работы.

3.2. Цель работы.

3.3. Теоретическую часть.

3.4. Текст выполненных программ.

3.5. Вывод.

Контрольные вопросы

1. Приведите рекомендуемые схемы управления светодиодной индикацией.

2. Опишите механизм ввода данных с клавиатуры в лабораторном стенде на основе микроконтроллеров семейства ATmega8515. Приведите рекомендуемые функциональные схемы ввода данных с клавиатуры.

3. Опишите механизм вывода данных на жидкокристаллический индикатор и приведите схему подключения ЖК-модуля к системной шине.

ЛАБОРАТОРНАЯ РАБОТА №4

Реализация системы управления на 8-разрядных микроконтроллерах ATmega8515: работа систем в реальном масштабе времени

Цель работы:

Изучение цифровой системы управления процессоров ATmega8515: функционирование таймерного модуля в МК ATmega8515, получение практических навыков в организации работы системы в реальном масштабе времени.

Используемое оборудование:

- персональная ЭВМ, совместимая с IBM PC.

Используемое программное обеспечение:

- операционная система Windows XP;
- интегрированная среда программирования AVR Studio.

1. Краткое описание работы

Для управления объектами в реальном масштабе времени МК используют различные таймерные блоки, которые могут определять время поступления сигналов от объектов управления и формировать выходные управляющие сигналы в заданные моменты времени. В МК семейства ATmega8515 для этих целей служит специальный модуль таймер-счетчик (*General Purpose Timer/Counter*) предназначенный для формирования запроса прерывания при истечении заданного интервала времени (режим таймера) или свершении заданного числа событий (режим счетчика). МК семейства AVR могут иметь от одного до трех 8- или 16- разрядных таймеров/счетчиков общего назначения.

Таймер/счетчик общего назначения может выполнять дополнительные функции:

- функцию захвата;
- функцию сравнения;
- функцию широтно-импульсной модуляцией (ШИМ);
- функцию счета реального времени.

Функция захвата заключается в запоминании кода, сформированного в базовом счетчике, в специальном регистре захвата при изменении значения определенного внешнего или внутреннего сигнала:

- регистр захвата таймера / счетчика Т/С1 (младший байт) – *ICR1L*;
- регистр захвата таймера/счетчика Т/С1 (младший байт) – *ICR1H*.

Функция сравнения заключается в изменении значения сигнала на определенном выходе МК при совпадении кода, формируемого в базовом счетчике, с кодом в специальном регистре сравнения:

- В таймер Т/С1 (младший байт) – *OCR1BL*;
- В таймер Т/С1 (старший байт) – *OCR1BH;*
- А таймер Т/С1 (младший байт) – *OCR1AL*;
- А таймер Т/С1 (старший байт) – *OCR1AH*.

Функция ШИМ заключается в формировании на определенном выходе МК импульсной последовательности с заданными периодом повторения и длительностью импульсов. Для формирования ШИМ сигнала предназначены следующие регистры:

- регистр управления В таймера/счетчика Т/С1 – *TCCR1B*;
- регистр управления А таймера/счетчика Т/С1 – *TCCR1A*;
- счетный регистр таймера/счетчика Т/С0 – *TCNTO*;
- регистр управления таймера/счетчика Т/С0 – *TCCRO*.

Функция счета реального времени (*Real Time Clock*) реализуется в таймере-счетчике при использовании дополнительного внутреннего генератора с внешним кварцевым резонатором с частотой 32768 Гц (часовой кварц). При этом параметры процессов в таймере-счетчике с высокой точностью привязаны к единице измерения реального времени – секунде. Счетный регистр таймера/счетчика T/C1 (младший байт) – *TCNT1L*, счетный регистр таймера/счетчика T/C1 (старший байт) – *TCNT1H*.

Таймер может генерировать прерывание при переполнении, то есть переходе из максимального значения в нулевое, и при совпадении значений регистров TCNTn и OCRn.

Разрешение и запрет прерываний от таймеров МК осуществляется соответствующими битами регистра TIMSK (рис. 12). Единичное значение бита разрешает соответствующее прерывание, нулевое значение бита – запрещает (маскирует) его. Такое управление реакцией МК на прерывания называется маскированием прерываний, а значение соответствующего регистра – маской прерываний. Для того чтобы МК реагировал на возникновение немаскированных прерываний необходимо, чтобы бит I регистра SREG был установлен в «1».

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

Рис. 12. Регистр TIMSK микроконтроллера

OCIE2 – прерывание по совпадению таймера T2;

TOIE2 – прерывание по переполнению таймера T2.

TICIE1 – прерывание по захвату таймера T1.

OCIE1A – прерывание по совпадению таймера T1;

OCIE1B – прерывание по совпадению таймера T1;

TOIE1 – прерывание по переполнению таймера T1.

OCIE0 – прерывание по совпадению таймера T0;

TOIE0 – прерывание по переполнению таймера T0.

Регистр TIFR отвечает за возникновение прерываний.

При наступлении события соответствующий флаг в регистре TIFR (рис. 13) устанавливается в единичное состояние. Расположение битов флагов в регистре TIFR аналогично расположению соответствующих битов разрешения прерывания в регистре TIMSK.

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

Рис. 13. Регистр TIFR микроконтроллера

При запуске программы обработки прерывания он аппаратно сбрасывается в 0.

8-битный таймер-счетчик Т0 реализованный в МК семейства ATmega8515 позволяет осуществлять:

- измерение промежутков времени;
- подсчет внешних событий;
- преобразование выходного сигнала контроллера в сигнал с (ШИМ).

Структурная схема представлена на рис. 14.

После подачи напряжения питания в регистре TCNTn находится нулевое значение. Регистр TCNTn изменяет свое значение на 1 с приходом каждого тактового сигнала. Тактовый сигнал может быть внутренним (сигнал с генератора, поступающий через предварительный делитель) или внешним, поступающим на соответствующий вход МК PB0/T0 (вывод 1).

Счетный регистр таймера-счетчика TCNT0 входит в состав основного блока модуля – блока реверсивного счетчика. В зависимости от режима работы модуля содержимое счетного регистра сбрасывается, инкрементируется или декрементируется по каждому импульсу тактового сигнала таймера-счетчика. Независимо от того, присутствует тактовый сигнал или нет, регистр доступен в любой момент времени как для чтения, так и для записи. Любая

операция записи в счетный регистр блокирует работу блока сравнения на время одного периода тактового сигнала таймера-счетчика.

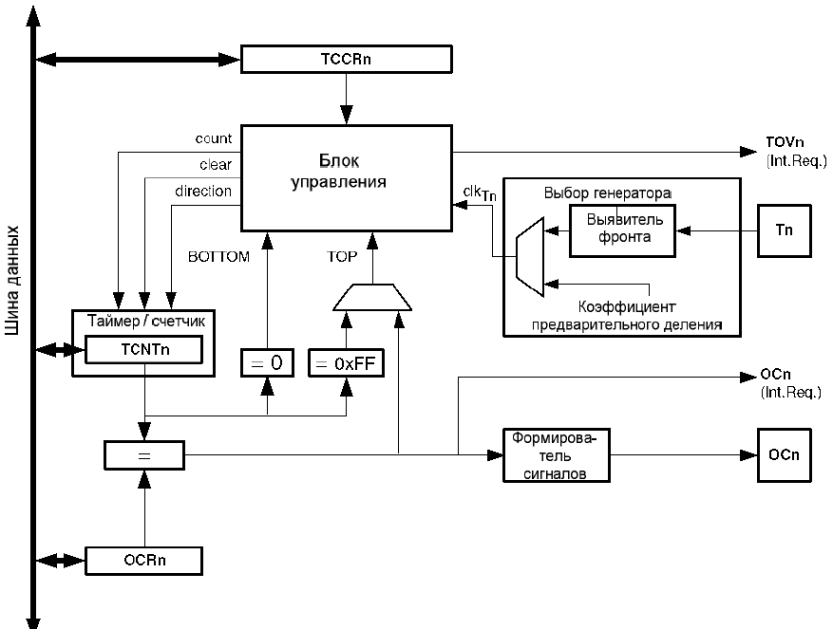

Рис. 14. Структурная схема таймера-счетчика T0

Регистр сравнения OCR0 входит в состав блока сравнения модуля. Во время работы таймера-счетчика производится непрерывное (в каждом такте) сравнение этого регистра с регистром TCNT0. В случае равенства содержимого этих регистров в следующем такте устанавливается флаг OCF0 в соответствующем регистре флагов и генерируется прерывание (если оно разрешено). При совпадении значений регистров может меняться состояние вывода OC0/PB0 МК.

Управление таймером-счетчиком T0 осуществляет регистр TCCR0 (рис. 15):

FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
7	6	5	4	3	2	1	0	

Рис. 15. Регистр TCCR0 микроконтроллера

FOC0 – принудительное изменение состояния вывода OC0/PB0 (вывод 4), в режиме ШИМ игнорируется и должен быть равен 0.

WGM01:WGM00 – режим работы таймера:

00 – нормальный режим;

01 – фазовый ШИМ;

10 – сброс при совпадении значений регистров TCNT0 и OCR0;

11 – быстрый ШИМ.

COM01:COM00 – подключение вывода OC0/PB0 (при единичном FOC0):

В режиме сброса при совпадении:

00 – вывод OC0/PB0 отключен и работает как линия порта ввода-вывода;

01 – переключает OC0/PB0 в противоположное состояние при совпадении значений регистров TCNT0 и OCR0;

10 – сбрасывает OC0/PB0 в 0 при совпадении значений регистров TCNT0 и OCR0;

11 – устанавливает OC0/PB0 в «1» при совпадении значений регистров TCNT0 и OCR0.

В режиме фазового ШИМа:

00 – вывод OC0/PB0 отключен и работает как линия порта ввода-вывода;

01 – зарезервирован;

10 – сбрасывает OC0/PB0 в «0» при совпадении значений регистров TCNT0 и OCR0 при счете на увеличение, устанавливает OC0/PB0 в «1» при счете на уменьшение;

11 – устанавливает OC0/PB0 в «1» при совпадении значений регистров TCNT0 и OCR0 при счете на увеличение, сбрасывает OC0/PB0 в «0» при счете на уменьшение.

В режиме быстрого ШИМа:

00 – вывод OC0/PB0 отключен и работает как линия порта ввода-вывода;

01 – зарезервирован;

10 – сбрасывает OC0/PB0 при совпадении значений регистров TCNT0 и OCR0, устанавливает OC0/PB0 при TCNT0=TOP;

11 – устанавливает OC0/PB0 при совпадении значений регистров TCNT0 и OCR0, сбрасывает OC0/PB0 при TCNT0=TOP.

CS02:CS00 – управление предварительным делителем:

000 – таймер отключен;

001 – коэффициент предварительного деления равен 1;

010 – коэффициент предварительного деления равен 8;

011 – коэффициент предварительного деления равен 64;

100 – коэффициент предварительного деления равен 256;

101 – коэффициент предварительного деления равен 1024;

110 – подключен внешний тактовый сигнал T0, активен передний фронт;

111 – подключен внешний тактовый сигнал T0, активен задний фронт.

Простейшим режимом работы таймера является его нормальный режим, при котором таймер работает как суммирующий счетчик, считающий входные импульсы. При переполнении 8-битной разрядной сетки значение счетного регистра TCNT0 сбрасывается. При этом устанавливается бит прерывания таймера TOV0.

Режим сброса при совпадении значений регистров TCNT0 и OCR0. Значение счетного регистра TCNT0 сбрасывается при равенстве его содержимого содержимому OCR0. Таким образом,

регистр OCR0 задает максимальную величину, до которой считает таймер. Бит прерывания OCF0 устанавливается каждый раз при совпадении значений TCNT0 и OCR0. Если значение регистра OCR0 меньше значения TCNT0, то счетный регистр сбрасывается при переполнении, после чего считает до значения, равного OCR0. Данный режим может использоваться для генерации меандровых импульсов с частотой, равной:

$$f = \frac{f_{osc}}{2 \cdot N \cdot (1 + OCR0)},$$

где N – коэффициент предварительного деления для таймера.

Быстрый ШИМ представляет собой генератор высокочастотного ШИМ сигнала с частотой:

$$f = \frac{f_{osc}}{256 \cdot N}$$

В общем случае частота быстрого ШИМ может быть в 2 раза выше, чем частота фазового ШИМ. Счетный регистр TCNT0 считает от 0 до максимального значения, после чего сбрасывается. Быстрый ШИМ может работать в неинвертирующем и инвертирующем режимах. В неинвертирующем режиме при совпадении значений TCNT0 и OCR0 состояние вывода OC0 сбрасывается в 0, а при переполнении счетного регистра устанавливается в «1». В инвертирующем режиме при совпадении значений TCNT0 и OCR0 состояние вывода OC0 устанавливается в «1», а при переполнении счетного регистра сбрасывается в «0». Бит прерывания TOV0 устанавливается каждый раз при переполнении счетчика.

Фазовый ШИМ имеет большую разрешающую способность по сравнению с быстрым ШИМ. Счетчик считает попеременно в возрастающем и убывающем порядке: от 0 до максимума, затем – от максимума до нуля. В неинвертирующем режиме вывод OC0 устанавливается в «1» при совпадении TCNT0 и OCR0 при сум-

мирующем счете и сбрасывается в «0» при совпадении TCNT0 и OCR0 при вычитающем счете. В инвертирующем режиме вывод OC0 работает инверсно. Флаг прерывания TOV0 устанавливается каждый раз при достижении счетным регистром максимального значения.

16-битный таймер-счетчик T1 реализованный в МК семейства ATmega8515 позволяет осуществлять:

- 16-битный счетный регистр TCNT1;
- 16-битный регистр захвата ICR1;
- два 16-битных регистра сравнения OCR1A, OCR1B;
- два 8-битных регистра управления TCCR1A, TCCR1B.

Упрощенная структурная схема таймера-счетчика T1 представлена на рис. 16.

Каждый 16-битный регистр таймера-счетчика размещается в двух регистрах ввода-вывода, названия которых получаются добавлением к названию регистра буквы «H» (старший байт) и «L» (младший байт). Счетный регистр таймера-счетчика Tl TCNT1 размещается в регистрах TCNT1H:TCNT1L. Этот регистр входит в состав основного блока модуля – блока реверсивного счетчика. В зависимости от режима работы модуля содержимое счетного регистра сбрасывается, инкрементируется или декрементируется по каждому импульсу тактового сигнала таймера-счетчика. Независимо от того, присутствует тактовый сигнал или нет, регистр доступен в любой момент времени как для чтения, так и для записи. При этом любая операция записи в счетный регистр блокирует работу всех блоков сравнения на время одного периода тактового сигнала таймера-счетчика.

После подачи напряжения питания в регистре TCNT1 находится нулевое значение. Во время работы таймера-счетчика производится непрерывное (в каждом такте) сравнение этих регистров с регистром TCNT1. В случае равенства содержимого регист-

ра сравнения и счетного регистра, в следующем такте устанавливается флаг OCF1A/OCF1B, в регистре флагов TIFR и генерируется прерывание (если оно разрешено). Также при наступлении этого события может изменяться состояние вывода OC1A/OC1B микроконтроллера. Чтобы таймер-счетчик мог управлять состоянием какого-либо из этих выводов, соответствующий вывод должен быть сконфигурирован как выходной (соответствующий бит регистра DDRn должен быть установлен в «1»).

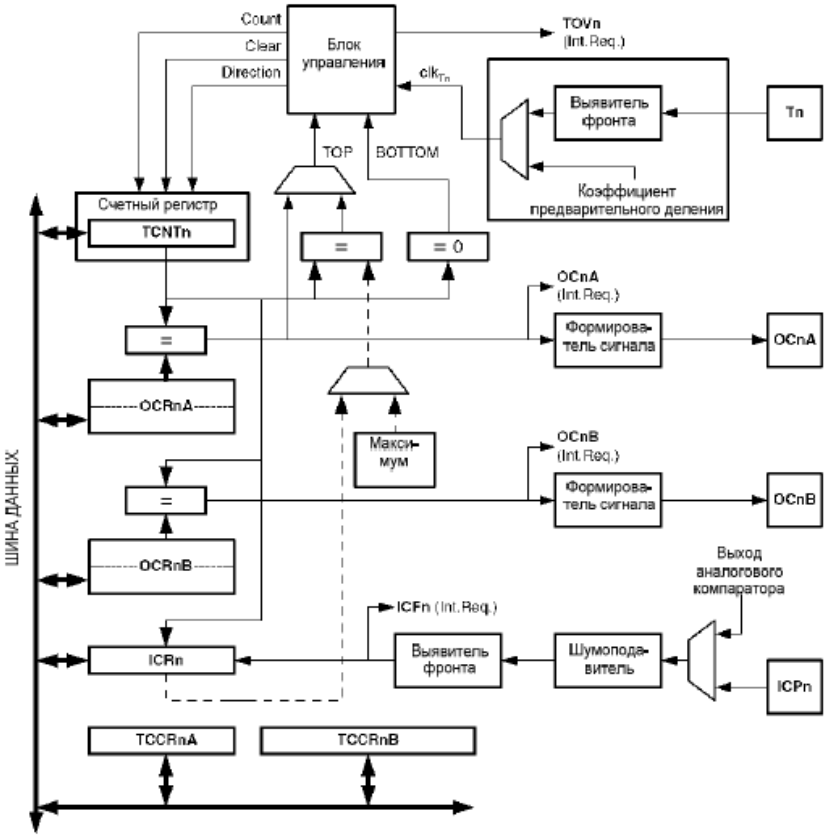

Рис. 16. Упрощенная структурная схема таймера T1 микроконтроллера

Особенностью работы блока сравнения в режимах, предназначенных для формирования ШИМ-сигналов, является двойная буферизация записи в регистры сравнения. Она заключается в том, что записываемое число на самом деле сохраняется в специальном буферном регистре. А изменение содержимого регистра сравнения происходит только при достижении счетчиком максимального значения.

Регистр захвата ICR1 входит в состав блока захвата, назначение которого – сохранение в определенный момент времени состояния таймера-счетчика в регистре захвата. 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 устанавливается флаг ICF1 регистра флагов TIFR и генерируется запрос на прерывание. Разрешение прерывания осуществляется установкой в «1» бита TICIE1 регистра маски.

Программно запись в регистр ICR1 возможна только в режимах, в которых регистр захвата определяет модуль счета таймера-счетчика. Вывод ICP в этих режимах отключен от МК, а функция захвата соответственно выключена.

Управление таймером-счетчиком T1 осуществляют регистры TCCR1A, TCCR1B (рис. 17):

COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
7	6	5	4	3	2	1	0	

Рис. 17. Регистры TCCR1A и TCCR1B микроконтроллера

COM1A1:COM1A0, COM1B1:COM1B0 – подключение выводов OC1A/PD5, OC1B/PE2 (при единичном FOC1A, FOC1B):

В режиме сброса при совпадении:

00 – OC1A/OC1B отключен и работает как линия порта ввода-вывода;

01 – переключает OC1A/OC1B в противоположное состояние при совпадении значений TCNT1A/TCNT1B и OCR1A/OCR1B;

10 – сбрасывает OC1A/OC1B в «0» при совпадении значений регистров TCNT1A/TCNT1B и OCR1A/OCR1B;

11 – устанавливает OC1A/OC1B в «1» при совпадении значений регистров TCNT1A/TCNT1B и OCR1A/OCR1B.

В режиме быстрого ШИМа:

00 – OC1A/OC1B отключен и работает как линия порта ввода-вывода;

01 – WGM13 = 0 – нормальная работа, OC1A/OC1B отключен;

WGM13=1 – переключает OC1A при совпадении, OC1B – зарезервирован;

10 – сбрасывает OC1A/OC1B при совпадении значений регистров TCNT1A/TCNT1B и OCR1A/OCR1B, устанавливает OC1A/OC1B при TCNT1A/TCNT1B = TOP;

11 – устанавливает OC1A/OC1B при совпадении значений регистров TCNT1A/TCNT1B и OCR1A/OCR1B, сбрасывает OC1A/OC1B при TCNT1A/TCNT1B = TOP.

В режиме фазового ШИМа:

00 – OC1A/OC1B отключен и работает как линия порта ввода-вывода;

01 – WGM13 = 0 – нормальная работа, OC1A/OC1B отключен; WGM13 = 1 переключает OC1A при совпадении, OC1B – зарезервирован;

10 – сбрасывает OC1A/OC1B при совпадении значений регистров TCNT1A/TCNT1B и OCR1A/OCR1B в режиме суммирования, устанавливает OC1A/OC1B при совпадении в режиме вычитания;

11 – устанавливает OC1A/OC1B при совпадении значений регистров TCNT1A/TCNT1B и OCR1A/OCR1B в режиме суммиро-

вания, сбрасывает OC1A/OC1B при совпадении в режиме вычитания.

FOC1A, FOC1B – принудительное изменение состояния вывода OC1A/OC1B, в режиме ШИМ игнорируется и должен быть равен «0».

WGM13:WGM10 – режим работы таймера:

0000 – нормальный режим;

0001 – 8-битный фазовый ШИМ;

0010 – 9-битный фазовый ШИМ;

0011 – 10-битный фазовый ШИМ;

0100 – сброс при совпадении значений регистров TCNT1A и OCR1A;

0101 – 8-битный быстрый ШИМ;

0110 – 9-битный быстрый ШИМ;

0111 – 10-битный быстрый ШИМ;

1000 – фазо-частотный ШИМ с модулем пересчета ICR1;

1001 – фазо-частотный ШИМ с модулем пересчета OCR1A;

1010 – фазовый ШИМ с модулем пересчета ICR1;

1011 – фазовый ШИМ с модулем пересчета OCR1A;

1100 – сброс при совпадении значений регистров TCNT1A и ICR1;

1101 – зарезервировано;

1010 – быстрый ШИМ с модулем пересчета ICR1;

1011 – быстрый ШИМ с модулем пересчета OCR1A;

ICNC1 – управление схемой подавления помех блока захвата. Если бит сброшен в «0», схема подавления помех выключена (захват производится по первому активному фронту).

Если бит установлен в «1», схема подавления помех включена и захват осуществляется только в случае четырех одинаковых выборок, соответствующих активному фронту сигнала ICES1 – выбор активного фронта сигнала захвата. Если бит сброшен в

«0», сохранение счетного регистра в регистре захвата осуществляется по спадающему фронту сигнала. Если бит установлен в «1», то сохранение счетного регистра в регистре захвата осуществляется по нарастающему фронту сигнала. Одновременно с сохранением счетного регистра устанавливается также флаг прерывания ICF1 регистра флагов.

CS12:CS10 – управление предварительным делителем:

000 – таймер отключен;

001 – коэффициент предварительного деления равен 1;

010 – коэффициент предварительного деления равен 8;

011 – коэффициент предварительного деления равен 64;

100 – коэффициент предварительного деления равен 256;

101 – коэффициент предварительного деления равен 1024;

110 – подключен внешний тактовый сигнал T1, активен передний фронт;

111 – подключен внешний тактовый сигнал T1, активен задний фронт.

Простейшим режимом работы таймера является его нормальный режим, при котором таймер работает как суммирующий счетчик, считающий входные импульсы. При переполнении 16-битной разрядной сетки значение счетного регистра TCNT1 сбрасывается. При этом устанавливается бит прерывания таймера TOV1.

Режим сброса при совпадении. Регистр OCR1A или ICR1 определяют разрешающую способность счетного регистра. Счетчик TCNT1 сбрасывается при достижении одного из значений регистров OCR1A или ICR1 (в зависимости от режима). Бит прерывания OCF1 устанавливается каждый раз при совпадении TCNT1 с соответствующим регистром OCR1A или ICR1. Данный режим может использоваться для генерации меандровых импульсов с частотой, равной:

$$f = \frac{f_{osc}}{2 \cdot N \cdot (1 + OCR1A[ICR1])},$$

где N – коэффициент предварительного деления для таймера.

Быстрый ШИМ представляет собой генератор высокочастотного ШИМ сигнала. Разрядность ШИМ может быть фиксирована (8, 9, 10 бит) или определяться одним из регистров OCR1A или ICR1 и может определяться как:

$$R_{PWM} = \frac{\lg(TOP + 1)}{\lg(2)},$$

Счетный регистр TCNT1 считает от 0 до максимального значения, после чего сбрасывается. Быстрый ШИМ может работать в неинвертирующем и инвертирующем режимах. В неинвертирующем режиме при совпадении значений TCNT1 и одного из регистров OCR1A/OCR1B состояние соответствующего вывода OC1A/OC1B сбрасывается в 0, а при переполнении счетного регистра устанавливается в «1». В инвертирующем режиме работа инверсна. Бит прерывания TOV1 устанавливается каждый раз при переполнении счетчика.

Фазовый ШИМ. Разрядность ШИМ может быть фиксирована (8, 9, 10 бит) или определяться одним из регистров OCR1A или ICR1 и может определяться аналогично быстрому ШИМ. В неинвертирующем режиме выводы OC1A/OC1B устанавливаются в «1» при совпадении TCNT1 и OCR1A/OCR1B при суммирующем счете и сбрасываются в «0» при совпадении TCNT1 и OCR1A/OCR1B при вычитающем счете. В инвертирующем режиме выводы OC1A/OC1B работает инверсно. Флаг прерывания TOV1 устанавливается каждый раз при достижении счетным регистром максимального значения.

Фазо-частотный ШИМ аналогичен фазовому. Отличие состоит в моменте перезаписи регистра OCR1x, участвующего в срав-

нении со счетным регистром. В фазовом режиме перезапись значений регистров OCR1x и модуля пересчета осуществляется в момент достижения максимального значения счетным регистром, а в фазо-частотном – в момент совпадения значений регистров OCR1x и TCNT1.

Управление электродвигателем. Одним из примеров практического использования функций таймер-счетчик реализованного в МК семейства ATmega8515 является управление вращением электродвигателя с помощью ШИМ - сигналов. Для этого на лабораторном стенде размещается электродвигатель постоянного тока и транзисторная схема, задающая ток в его обмотку. Управление двигателем осуществляется ШИМ - сигналами, поступающими с выходов:

- выход A таймера/счетчика T1;
- выход B таймера/счетчика T1.

Пользователь задает значения скважности сигналов в регистрах A таймера/счетчика T1, B таймера/счетчика T1. После инициализации на выходах формируется положительный перепад напряжения, а затем вычисляется время формирования последующих отрицательных и положительных перепадов.

На плате размещен электродвигатель постоянного тока типа. Для управления двигателем используются выход A, выход B двух каналов таймерного модуля, на которые выдаются сигналы с ШИМ - последовательность импульсов напряжения с заданной частотой и скважностью. Параметры ШИМ-сигналов - частота и длительность импульсов, задаются путем соответствующего программирования таймерного модуля в МК ATmega8515. ШИМ-сигналы с выходов таймерного модуля поступают на мостовую транзисторную схему, представленную на рис. 18. Эта схема обеспечивает необходимое значение тока, поступающего в обмотку электродвигателя.

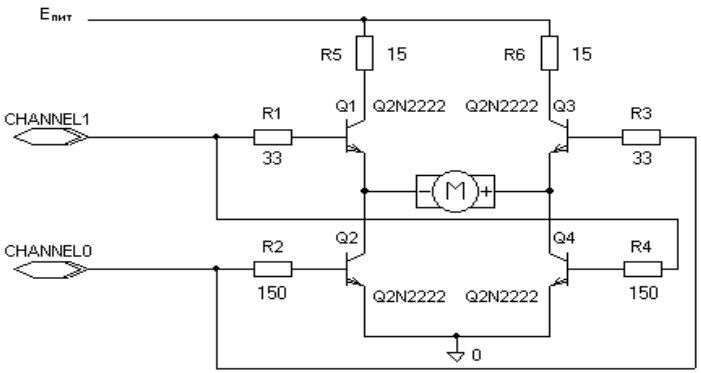

Рис.18. Схема включения электродвигателя

Величина тока в каждом плече мостовой схемы обратно пропорциональна скважности ШИМ-сигналов, поступающих на базу транзистора, включенного в этом плече:

I0 = k/Q0 - ток протекающий через транзисторы Q2,Q3;

I1 = k/Q1 - ток протекающий через транзисторы Q1,Q4.

где Q0, Q1 - скважность импульсов, поступающих с каналов A, B, соответственно, k - коэффициент пропорциональности, определяемый параметрами компонентов мостовой схемы. Ток Iд, протекающий в обмотке электродвигателя, равен разности токов ветвей:

Iд = I1-10 = k (Q1 - Q0)/Q1∗Q0.

Таким образом, направление протекания тока Iд и, соответственно, направление вращения ротора электродвигателя определяется относительным значением скважности ШИМ-сигналов, формируемых на выходах каналов A, B. Скорость вращения пропорциональна величине тока Iд, то-есть разности значений скважности (Q1-Q0).

Таким образом, с помощью соответствующего программирования таймерного модуля можно обеспечить различные режимы работы электродвигателя.

Далее представлен пример программы использования таймерного модуля на языке ассемблера для процессоров ATmega8515.

```
.include "8515def.inc"
.def temp = r16
ser temp
out DDRD, temp
ldi temp, low(ramend)
out spl, temp
ldi temp, high(ramend)
out sph, temp
init:

    rcall clear
    ldi temp, 0b11000001
    out TCCR1A, temp
    ldi temp, 0x01
    out TCCR1B, temp
    ldi r18, 0x01
    ldi r19, 0x01
    rjmp Timeloop

    rjmp time
time:
    inc r17
    cpi r17, 100
    brne Timeloop
    clr r17
    clr temp
    out TCCR1B, temp
    out TCCR0, temp
    cpse r18, r19
    rjmp init

    ldi temp, 0b00100001
    out TCCR1A, temp
```

```
        ldi temp, 0x01
        out TCCR1B, temp
        ldi r18, 0x01
        ldi r19, 0x02
        rjmp Timeloop

clear:
        clr temp
        out OCR1AH, temp

        out OCR1AL, temp
        out OCR1BH, temp
        out OCR1BL, temp
        out TCNT1H, temp
        out TCNT1L, temp
        out TCNT0, temp
        ret
Timeloop:
        ldi temp, 0xFF
        out 0x31, temp
        ldi temp, 0b00000101
        out TCCR0, temp
        ldi temp, 0b00000101
        sei
        out TIMSK, temp
        rjmp timeloop
```

2. Порядок выполнения работы

2.1. Ознакомиться с функционированием таймерного модуля, выполняемых микроконтроллеров семейства ATmega8515.

2.2. Подготовить лабораторный макет STK500 к работе, выполнив следующие операции:

- включить источник питания платы и инструментальный компьютер;
- запустить из рабочего каталога интегрированный отладчик AVR Studio.

2.3. Провести инициализацию таймерного модуля, установив необходимый режим его работы. Ввести в память контроллерной платы программу на языке ассемблера представленную выше.

2.4. Составить программу на языке ассемблера для реверсивного управления электродвигателем постоянного тока.

2.5. Результат выполненной работы в интегрированной среде программирования занести в отчет.

3. Содержание отчета

3.1. Наименование лабораторной работы.

3.2. Цель работы.

3.3. Теоретическую часть.

3.4. Текст выполненных программ.

3.5. Вывод.

Контрольные вопросы

1. Приведите структуру 8-битного таймера-счетчика T0 и 16-битного таймера-счетчика T1 реализованного в МК ATmega8515, приведите выполняемые функции и методику расчета длительности импульсов для реализации ШИМ-сигналов.

2. Приведите рекомендуемые функциональные схемы управления электродвигателем, соленоидом, реле и высоковольтной нагрузкой.

3. Приведите комментарии к программе управления электродвигателем в лабораторном стенде на основе МК ATmega8515.

ЛАБОРАТОРНАЯ РАБОТА №5

Реализация системы управления на 8-разрядных микроконтроллерах ATmega8515

Цель работы:

Получение практических навыков в организации цифровой системы управления микроконтроллеров ATmega8515.

Используемое оборудование:

- персональная ЭВМ, совместимая с IBM PC.

Используемое программное обеспечение:

- операционная система Windows XP;
- интегрированная среда программирования AVR Studio.

1. Порядок выполнения работы

1.1. Подготовить лабораторный макет к работе, выполнив следующие операции:

- включить источник питания платы и инструментальный компьютер;
- запустить из рабочего каталога интегрированный отладчик AVR Studio.

1.2. Составить программу на языке ассемблера по заданию преподавателя.

1.3. Провести инициализацию платы STK500, установив необходимый режим его работы. Ввести в память контроллерной платы программу на языке ассемблера.

1.4. Результат выполненной работы в интегрированной среде программирования занести в отчет.

2. Содержание отчета

2.1. Наименование лабораторной работы.

2.2. Цель работы.

2.3. Теоретическую часть.

2.4. Текст выполненных программ.

2.5. Вывод.

Библиографический список

1. Болл Стюарт Р. Аналоговые интерфейсы микроконтроллеров. – М.: Издательский дом «Додэка-XXI», 2007. – 360 с.

2. Трамперт, В. AVR-RISC микроконтроллеры.: Пер. с нем. – К.: "МК-Пресс", 2006. – 464 с.

3. Евстифеев, А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы «ATMEL» – М.: Издательский дом «Додэка-XXI», 2004. – 560 с.

4. Гребнев, В.В. Микроконтроллеры семейства AVR фирмы Atmel. – М.: ИП РадиоСофт, 2002. – 176 с.

5. Мортон Дж. Микроконтроллеры AVR. Вводный курс / Пер. с англ. – М.: Издательский дом «Додэка-XX1», 2006. – 272 с.

6. Трамперт, В. Измерение, управление и регулирование с помощью AVR-микроконтроллеров.: Пер. с нем. – К.: "МК-Пресс", 2006. – 208 с.

7. Баранов, В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы. – М.: Издательский дом «Додэка-XXI», 2004. – 288 с.

8. Хартов, В.Я. Микроконтроллеры AVR. Практикум для начинающих. -М: Изд-во МГТУ им. Н.Э. Баумана, 2007. - 240 с:

ПРИЛОЖЕНИЕ

Краткая информация по 8-разрядным RISC микроконтроллерам семейства AVR представлена в таблицах 1, 2, 3.

Таблица 1

Микроконтроллеры семейства TinyAVR

Тип	Напр. питания В	Такт. частота МГц	I/O	Flash	EEPROM	SRAM	Интер-фейсы	АЦП	Таймеры	ISP	Корпус
1	2	3	4	5	6	7	8	9	10	11	12
ATtiny11	2.7-5.5	6	6	1K	-	-	-	-	1x8bit	-	PDIP8 SOIC8
ATtiny12	1.8-5.5	6	6	1K	64	-	-	-	1x8bit	I	PDIP8 SOIC8
ATtiny13	1.8-5.5	20	6	1K	64	64	-	4x10bit	1x8bit 2xPWM	I	PDIP8 SOIC8
ATtiny15L	2.7-5.5	6	6	1K	64	-	-	4x10bit	2x8bit	I	PDIP8 SOIC8
ATtiny2313	1.8-5.5	20	15	2K	128	128	SPI UART	-	1x8bit 1x16bit	I	PDIP20 SOIC20
ATtiny24	1,8…5,5	20	12	2K	128	128	USI 4xPWM RTC	8x10bit	1x8bit 1x16bit	S	PDIP14 MLF20 SOIC14
ATtiny25	2,7…5,5	20	32	2K	128	128	SPI UART	4x10bit	1x8bit 1x8bit high speed	I	PDIP8 SOIC8
ATtiny25	2,7…5,5	16	32	2K	128	128	SPI UART	4x10bit	1x8bit 1x8bit	I	SOIC8
ATtiny25V	1.8 - 5.5	10	32	2K	128	128	SPI UART	4x10bit	1x8bit 1x8bit	I	PDIP8 SOIC8
ATtiny26	2.7-5.5	16	16	1K	128	128	SPI UART	11x10bit	2x8bit	I	PDIP20 SOIC20 MLF32
ATtiny261	1.8-5.5	20	16	2K	128	128	PWM USI	11x10bit	1x8bit 1x16bit	I	PDIP20 SOIC20 MLF32
Ttiny461	1.8-5.5	20	16	4K	256	256	PWM USI	11x10bit	1x8bit 1x16bit	I	PDIP20 SOIC20 MLF32
ATtiny28L	1.8-5.5	4	20	2K	-	-	-	-	1x8bit	-	PDIP28 TQFP32 MLF32
ATtiny44	1,8…5,5	20	12	4K	256	256	USI 4xPWM RTC	8x10bit	1x8bit 1x16bit	S	PDIP14 MLF20 SOIC14

ATtiny45	2,7…5,5	20	32	4K	256	256	SPI UART	4x10bit	1x8bit 1x8bit high speed	I	PDIP8 SOIC8
ATtiny45V	1.8 - 5.5	10	32	4K	256	256	SPI UART	4x10bit	1x8bit 1x8bit high speed	I	PDIP8 SOIC8
ATtiny84	1,8…5,5	20	12	8K	512	512	USI 4xPWM RTC	8x10bit	1x8bit 1x16bit	S	PDIP14 MLF20 SOIC14
ATtiny85	2,7…5,5	20	32	8K	512	256	SPI UART	4x10bit	1x8bit 1x8bit high speed	I	PDIP8 SOIC8
ATtiny85V	1.8 - 5.5	10	32	8K	512	256	SPI UART	4x10bit	1x8bit 1x8bit high speed	I	PDIP8 SOIC8
ATtiny861	1.8-5.5	20	16	8K	256	256	PWM USI	11x10bit	1x8bit 1x16bit	I	PDIP20 SOIC20 MLF32

Таблица 2

Классические AVR-микроконтроллеры

Тип	Напр. питания, В	Такт. Частота,	I/O	Flash	EEPROM	SRAM	Интер-фейсы	АЦП	Таймеры	Корпус
AT90PWM1	2.7-5.5	16	19	8K	0.5	512	SPI	8x10bit	1x8bit 1x16bit	SO24
AT90PWM2	2.7-5.5	16	53	8K	512	512	SPI	8x10bit	2	SO24
AT90PWM3	2.7-5.5	16	53	8K	512	512	SPI	11x10bit	2	SO32, QFN32
AT90S1200	2.7-6.0 4.0-6.0	4 12	15	1K	64	-	-	-	1x8bit	DIP20 SO20 SSOP20
AT90S2313	2.7-6.0 4.0-6.0	4 10	15	2K	128	128	UART	-	1x8bit 1x16bit	DIP20 SO20
AT90LS2323	2.7-6.0	4	3	2K	128	128	-	-	1x8bit	DIP8 SO8
AT90S2323	4.0-6.0	10	3	2K	128	128	-	-	1x8bit	DIP8 SO8
AT90LS2343	2.7-6.0	4	5	2K	128	128	-	-	1x8bit	DIP8 SO8
AT90S2343	4.0-6.0	10	5	2K	128	128	-	-	1x8bit	DIP8 SO8
AT90LS4433	2.7-6.0	4	20	4K	256	128	UART SPI	6x10bit	1x8bit 1x16bit	DIP28 TQFP32
AT90S4433	4.0-6.0	8	20	4K	256	128	UART SPI	6x10bit	1x8bit 1x16bit	DIP28 TQFP32
AT90LS8515	2.7-6.0	4	32	8K	512	512	UART SPI	-	2x8bit 1x16bit	DIP40 TQFP44 PLCC44
AT90S8515	4.0-6.0	8	32	8K	512	512	UART SPI	-	2x8bit 1x16bit	DIP40 TQFP44

										PLCC44
AT90LS8535	2.7-6.0	4	32	8K	512	512	UART SPI	8x10bit	2x8bit 1x16bit	DIP40 TQFP44 PLCC44
AT90S8535	4.0-6.0	8	32	8K	512	512	UART SPI	8x10bit	2x8bit 1x16bit	DIP40 TQFP44 PLCC44

Таблица 3

Микроконтроллеры семейства MegaAVR

Тип	Напр. питания, В	Такт. Частота,	I/O	Flash	EEPROM	SRAM	Интер-фейсы	АЦП	Тайме-ры	Корпус
ATmega406	4.0 - 25	1	18	40K	512	2K	JTAG TWI	10x12bit 1x18bit	1x8bit 1x16bit	LQFP48
ATmega48	1.8-5.5	20	23	4K	256	512	UART SPI I^2C	6x10bit 2x8bit	2x8bit 1x16bit	DIP28 TQFP32 MLF32
ATmega88	1.8-5.5	20	23	8K	512	1k	UART SPI I^2C	6x10bit 2x8bit	2x8bit 1x16bit	DIP28 TQFP32 MLF32
ATmega168	1.8-5.5	20	23	16K	512	1k	UART SPI I^2C	6x10bit 2x8bit	2x8bit 1x16bit	DIP28 TQFP32 MLF32
ATmega8	2.7-5.5	16	23	8K	512	1k	UART SPI	8x10bit	2x8bit 1x16bit	DIP28 TQFP32 MLF32
ATmega16	2.7-5.5	16	32	16K	512	1k	UART SPI	8x10bit	2x8bit 1x16bit	DIP40 TQFP44 MLF44
ATmega32	2.7-5.5	16	32	32K	1K	2K	UART SPI	8x10bit	2x8bit 1x16bit	DIP40 TQFP44 MLF44
ATmega64	2.7-5.5	16	53	64K	2K	4K	2xUART SPI	8x10bit	2x8bit 2x16bit	TQFP64 MLF64
ATmega640	1,8…5,5 4,5…5,5	8 16	86	64K	4K	8K	4xUART JTAG SPI	16x10bit	2x8bit 4x16bit	TQFP100
ATmega128	2.7-5.5	16	53	128K	4K	4K	2xUART SPI	8x10bit	2x8bit 2x16bit	TQFP64 MLF64
ATmega1280	1,8…5,5 4,5…5,5	8 16	86	128K	4K	8K	4xUART JTAG SPI	16x10bit	2x8bit 4x16bit	TQFP100
ATmega1281	1,8…5,5 4,5…5,5	8 16	54	128K	4K	8K	2xUART JTAG SPI	8x10bit	2x8bit 4x16bit	TQFP64
AT90CAN32	2.7-5.5	16	53	32K	1K	2048	UART	8x10bit	2x8bit	MLF 64

							JTAG CAN USART		2x16bit	LQFP 64
AT90CAN64	2.7-5.5	16	53	64K	2K	4K	UART JTAG CAN USART	8x10bit	2x8bit 2x16bit	MLF 64 LQFP 64
AT90CAN128	2.7-5.5	16	53	128K	4K	4K	2xUART SPI CAN	8x10bit	2x8bit 2x16bit	TQFP64 MLF64
ATmega103	4.0-5.5	6	48	128K	4K	4K	UART SPI	8x10bit	2x8bit 2x16bit	TQFP64
ATmega161	2.7-5.5	8	35	16K	512	1K	2xUART SPI	-	2x8bit 1x16bit	DIP40 TQFP44
ATmega162	1.8-5.5	16	35	16K	512	1K	2xUART SPI	-	2x8bit 1x16bit	DIP40 TQFP44 MLF44
ATmega163L	2.7-5.5	8	32	16K	512	1K	UART SPI	8x10bit	2x8bit 1x16bit	DIP40 TQFP44 MLF44
ATmega164P	1.8-5.5	16	32	16K	512 K	1024	2xUART SPI+US ART TWI	8x10bit	2x8bit 1x16bit	MLF44 PDIP40 TQFP44
ATmega165	1.8-5.5 2.7-5.5	8 16	53	16K	512	1K	UART SPI JTAG PWM	8x10bit	2x8bit 1x16bit	TQFP64 MLF64
ATmega165P	1.8-5.5	16	54	16K	0.5	1024	UART SPI+USI 4PWM	8x10bit	2x8bit 1x16bit	MLF64 TQFP64
ATmega169	1.8-3.6	4	53 4x25 LCD	16K	512	1K	UART SPI	8x10bit	2x8bit 1x16bit	TQFP64
ATmega8515	2.7-5.5	16	35	8K	512	512	UART SPI	-	2x8bit 1x16bit	PDIP40 PLCC44 TQFP,
ATmega8535	2.7-5.5	16	32	8K	512	512	UART SPI	8x10bit	2x8bit 1x16bit	PDIP40 PLCC44 TQFP MLF
ATmega2560	1,8…5,5 4,5…5,5	8 16	86	256K	4K	8K	2xUART JTAG SPI	16x10bi t	2x8bit 4x16bit	TQFP100
ATmega2561	1,8…5,5 4,5…5,5	8 16	54	256K	4K	8K	2xUART JTAG SPI	8x10bit	2x8bit 4x16bit	TQFP64
ATmega324P	1.8-5.5	20	32	32K	1K	2048	2xUART SPI+US ART	8x10bit	2x8bit 1x16bit	MLF44 PDIP40 TQFP44

							TWI			
ATmega325	1.8-5.5	16	53	32K	1K	2K	UART SPI	8x10bit	2x8bit 1x16bit	TQFP MLF
ATmega3250	1.8-5.5	16	68	32K	1K	2K	UART SPI	8x10bit	2x8bit 1x16bit	TQFP MLF
ATmega325P	1.8-5.5	20	54	32K	1K	2048	UART SPI	8x10bit	2x8bit 1x16bit	MLF64 TQFP64
ATmega3250P	1.8-5.5	20	54	32K	1K	2048	UART SPI	8x10bit	2x8bit 1x16bit	TQFP100
ATmega329P	1.8-5.5	16	54	32K	1K	2048	JTAG SPI	8x10bit	2x8bit 1x16bit	MLF64 TQFP64
ATmega3290P	1.8-5.5	16	54	32K	1K	2048	JTAG SPI	8x10bit	2x8bit 1x16bit	TQFP100
ATmega644P	1.8-5.5	20	32	64K	2K	4096	2xUART SPI+US ART TWI	8x10bit	2x8bit 1x16bit	MLF44 PDIP40 TQFP44
ATmega645	1.8-5.5	16	53	64K	2K	4K	UART SPI	8x10bit	2x8bit 1x16bit	TQFP MLF
ATmega6450	1.8-5.5	16	68	64K	2K	4K	UART SPI	8x10bit	2x8bit 1x16bit	TQFP MLF
ATmega644	1.8-5.5 2.7-5.5	10 20	32	64K	2K	4K	UART SPI TWI PWM JTAG	8x10bit	2x8bit 1x16bit	PDIP40 TQFP44 MLF44
ATmega329	1.8-5.5	16	53 LCD 4x25	32K	1K	2K	UART SPI	8x10bit	2x8bit 1x16bit	TQFP MLF
ATmega3290	1.8-5.5	16	68 LCD 4x40	32K	1K	2K	UART SPI	8x10bit	2x8bit 1x16bit	TQFP MLF
ATmega649	1.8-5.5	16	53 LCD 4x25	64K	2K	4K	UART SPI	8x10bit	2x8bit 1x16bit	TQFP MLF