МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН
ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Факультет «Телекоммуникационных технологий»

Кафедра «Сети и системы передачи данных»

МИКРОПРОЦЕССОРЫ

(часть 2)

Методические указания по выполнению лабораторных работ

Ташкент 2015

ВВЕДЕНИЕ

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

Микроконтроллерные системы изучаются на примере одних из наиболее популярных в настоящее время в мире микроконтроллеров AVR фирмы «Atmel». Дается краткое описание архитектуры и системы команд AVR-микроконтроллеров, ассемблерных директив и особенностей построения ассемблерных программ и основных возможностей AVR-Studio.

Лабораторные работы по программированию AVR-микроконтроллеров рассчитаны на освоение их основных программно-аппаратных средств: системы команд, портов ввода-вывода, системы прерываний, таймеров/счетчиков, аналого-цифрового преобразователя, а также особенностей построения микроконтроллерных систем. В каждой лабораторной работе дается необходимый теоретический материал. Кроме лекционного курса учебное пособие включает руководство к выполнению трех компьютерных лабораторных работ по одноименной дисциплине. Для их реализации используются программные средства отладки, свободно распространяемые фирмой Atmel (отладчик AVR Studio, компилятор CVAVR и симулятор VMLAB).Для облегчения процесса написания студентами индивидуальных программ приводится ряд примеров составления аналогичных программ.

Лабораторная работа № 1Микроконтроллер ATTINY15L

Цель работы. Целью лабораторной работы является отладка прикладных программ для микроконтроллера AVR семейства Tiny с помощью персонального компьютера и программных средств отладки.

ATtiny15L является 8-разрядным микроконтроллером с низким уровнем энергопотребления, основанным на AVR RISC-архитектуре. Благодаря выполнению высокопроизводительных инструкций за один период тактового сигнала ATtiny15L достигает производительности, приближающейся к уровню 1 MIPS на МГц, обеспечивая разработчику возможность оптимизировать уровень энергопотребления в соответствии с необходимым быстродействием. Ядро AVR содержит мощный набор инструкций (90 команд) и 32 рабочих регистра общего назначения. Все 32 регистра напрямую подключены к арифметико-логическому устройству (АЛУ), что обеспечивает доступ к двум независимым регистрам при выполнении одной инструкции за один такт. Данная архитектура позволяет повысить быстродействие вплоть до 10 раз по сравнению со стандартными микроконтроллерами CISC (рис. 1.1).

ATtiny15L имеет: 1 Кбайт Flash-памяти программ (512 16-разрядных ячеек), 64 байта энергонезависимой памяти данных EEPROM, 6 линий I/O общего назначения, 32 регистра общего назначения, два 8-разрядных универсальных таймера/счетчика, один с высокоскоростным выходом с ШИМ, встроенные генераторы, внутренние и внешние прерывания, программируемый сторожевой таймер, аналоговый компаратор, 4-канальный 10-раз-рядный АЦП, а также три программно выбираемых режима экономии энергопотребления. Режим ожидания «IdleMode» останавливает CPU, но позволяет функционировать АЦП, аналоговому компаратору, таймеру/счетчикам и системе прерываний. Режим подавления шумов АЦП

обеспечивает высокопрецизионные АЦП-измерения путем остановки CPU и сохранения работоспособности АЦП.

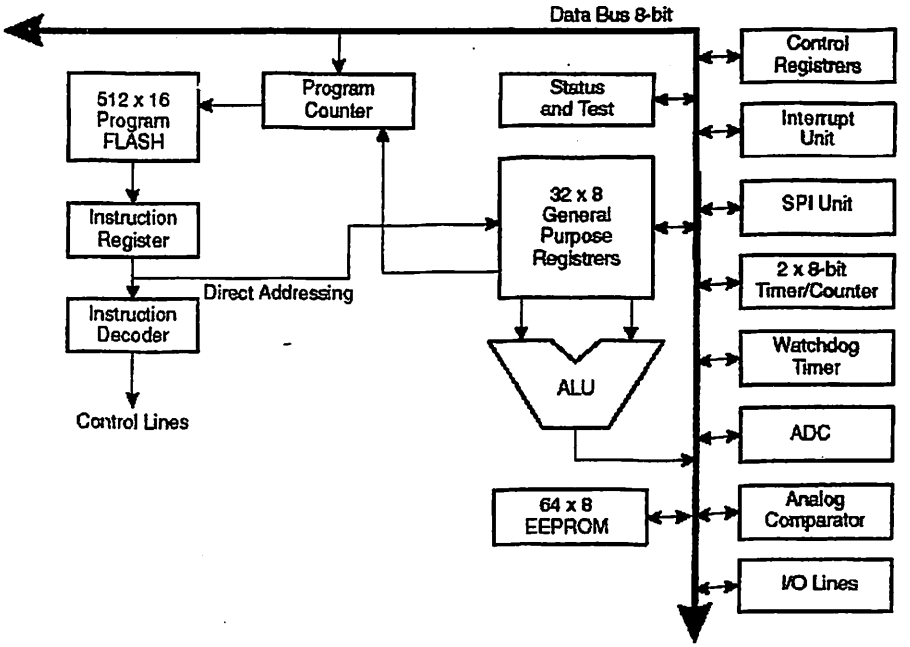


Рис. 1.1 — Структурная схема ATtiny15L

Режим экономии энергопотребления «PowerDown» сохраняет содержимое регистров, но останавливает тактовые генераторы, отключая все остальные функции микроконтроллера, вплоть до следующего внешнего прерывания или до аппаратной инициализации. Функция активации, или прерывания при смене логического уровня на линии порта, позволяет ATtiny15L быть высокочувствительным к внешним событиям, при сохранении минимального уровня энергопотребления при нахождении в режимах экономии энергопотребления.

Устройство производится с применением технологии энергонезависимой памяти с высокой плотностью размещения,

разработанной в корпорации Atmel. Благодаря совмещению усовершенствованного _8-разрядного RISC CPU с Flash-памятью с поддержкой внутрисистемного программирования на одном кристалле получился высокопроизводительный микроконтроллер ATtiny15L, обеспечивающий гибкое и экономически высокоэффективное решение для многих приложений встраиваемых систем управления, особенно в случае применения в зарядных устройствах, системах балластного освещения и во всех типах приложений, использующих интеллектуальные датчики.

Напряжение питания от 2,7 В до 5,5 В. Внутренняя тактовая частота 1,6 МГц. Коммерческий и индустриальный диапазоны эксплуатационных температур. Корпус имеет 8 выводов (рис. 1.2). Альтернативные функции линий порта В указаны в табл. 1.1.

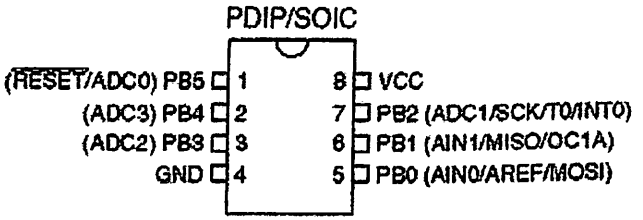


Рис. 1.2 — Расположение выводов ATtiny15L

Таблица 1.1 — Дополнительные функции линий порта В

Линия PB	Альтернативная функция
PB0	MOSI (Вход данных при программировании) AREF (Вход опорного напряжения для АЦП) AIN0 (Неинвертирующий вход аналогового компаратора)
PB1	MISO (Выход данных при программировании) OC1A (Выход таймера/счетчика T1 в режиме ШИМ) AIN1 (Инвертирующий вход аналогового компаратора)
PB2	SCK (Вход тактового сигнала при программировании) INT0 (Вход внешнего прерывания) ADC1 (Вход АЦП) T0 (Вход внешнего тактового сигнала таймера/счетчика T0)

PB3	ADC2 (Вход АЦП)
PB4	ADC3 (Вход АЦП)
PB5	RESET (Вход сброса) ADC0 (Вход АЦП)

Обращение к порту B производится с помощью регистров PORTB (регистр данных порта B), DDRB (регистр направления порта B), PINB (регистр выводов порта B). Формат этих регистров приведен в табл. 1.2.

Таблица 1.2 — Формат регистров порта B

7	6	5	4	3	2	1	0
–	–	–	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
–	–	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
–	–	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0

Максимальная нагрузочная способность выводов PB4...PB0 составляет 20 мА, а вывода PB5 — 12 мА. Вывод PB5 может функционировать либо как вход, либо как выход с открытым стоком (на выводе может присутствовать только сигнал логического 0).

При установке разряда DDBn в 1 соответствующийn-вывод порта является выходом, при сбросе в 0 — входом. Для линий, сконфигурированных как входные, имеется возможность подключения внутренних подтягивающих резисторов сопротивлением 35...120 кОм между входом и шиной питания Vcc. Если разряд PUD (6-ой бит регистра MCUCR) установлен в 1, подтягивающие резисторы отключаются от всех линий порта. Для подключения подтягивающего резистора необходимо сбросить разряд PUD и записать 1 в соответствующий разряд регистра PORTB (вывод PB5 не имеет внутреннего подтягивающего резистора).

Полный перечень регистров ввода/вывода микроконтроллера ATtiny15L приведен в табл. 1.3. Младшие адреса памяти программ отведены под таблицу векторов прерываний (табл. 1. 4).

Таблица 1.3 — Регистры ввода/вывода (адрес, название и функция)

$3F	SREG	StatusRegister
$3B	GIMSK	General Interrupt Mask Register
$3A	GIFR	General Interrupt Flag Register
$39	TIMSK	Timer/Counter Interrupt Mask Register
$38	TIFR	Timer/Counter Interrupt Flag Register
$35	MCUCR	MCU Control Register
$34	MCUSR	MCU Status Register
$33	TCCR0	Timer/Counter0 Control Register
$32	TCNT0	Timer/Counter0 (8-bit)
$31	OSCCAL	Oscillator Calibration Register
$30	TCCR1	Timer/Counter1 Control Register
$2F	TCNT1	Timer/Counter1 (8-bit)
$2E	OCR1A	Timer/Counter1 Output Compare Register A
$2D	OCR1B	Timer/Counter1 Output Compare Register B
$2C	SFIOR	Special Function I/O Register
$21	WDTCR	Watchdog Timer Control Register
$1E	EEAR	EEPROM Address Register
$1D	EEDR	EEPROM Data Register
$1C	EECR	EEPROM Control Register
$18	PORTB	Data Register, Port B
$17	DDRB	Data Direction Register, Port B
$16	PINB	Input Pins, Port B
$08	ACSR	Analog Comparator Control and Status Register
$07	ADMUX	ADC Multiplexer Select Register
$06	ADCSR	ADC Control and Status Register

$05	ADCH	ADC Data Register High
$04	ADCL	ADC Data Register Low

Таблица 1.4 — Векторы прерываний (адрес, источник и описание)

$000	RESET	External Reset, Power-on Reset,
		Brown-out Reset, and Watchdog Reset
$001	INT0	External Interrupt Request 0
$002	I/O Pins	Pin Change Interrupt
$003	TIMER1, COMPA	Timer/Counter1 Compare Match A
$004	TIMER1, OVF	Timer/Counter1 Overflow
$005	TIMER0, OVF	Timer/Counter0 Overflow
$006	EE_RDY	EEPROM Ready
$007	ANA_COMP	Analog Comparator
$008	ADC	ADC Conversion Complete

Таймеры ATtiny15L

Таймер/счетчик T0 (счетный регистр TCNT0) может работать в режиме таймера (на вход поступают импульсы тактового сигнала микроконтроллера непосредственно или через предделитель) или в режиме счетчика внешних событий. Режим работы задают три младших разряда регистра управления TCCR0 (табл. 1.5). Другие разряды этого регистра не используются.

Таблица 1.5 — Выбор источника тактового сигнала для T0

CS02	CS01	CS00	Источник тактового сигнала
0	0	0	Таймер/счетчик остановлен
0	0	1	CK (тактовый сигнал микроконтроллера)
0	1	0	CK/8

0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	Вывод T0, инкремент счетчика производится по спадающему фронту импульсов
1	1	1	Вывод T0, инкремент счетчика производится по нарастающему фронту импульсов

При переходе таймера/счетчика из состояния $FF в состояние $00 устанавливается флаг TOV0 (табл. 1.6) и генерируется запрос на прерывание. Разрешение прерывания осуществляется установкой в 1 разряда TOIE0 (табл. 1.7) при условии, что флаг общего разрешения прерываний I регистра SREG также установлен в 1.

Таблица 1.6 — Формат регистра флагов прерываний TIFR

7	6	5	4	3	2	1	0
–	OCF1A	–	–	–	TOV1	TOV0	–

Таблица 1.7 — Формат регистра масок прерываний TIMSK

7	6	5	4	3	2	1	0
–	OCIE1A	–	–	–	TOIE1	TOIE0	–

В таймере/счетчике T1 возможность счета внешних импульсов отсутствует. Однако он может выполнять определенные действия при равенстве содержимого счетного регистра TCNT1 и регистров выходного сравнения OCR1A и OCR1B. Кроме того, он может работать как широтно-импульсный модулятор для генерирования сигнала с программируемой частотой и скважностью. Дальнейшее описание T1 приводится именно для режима ШИМ.

При работе таймера/счетчика в режиме ШИМ состояние счетного регистра изменяется от $00 до значения, находящегося в регистре OCR1B, после чего счетный регистр сбрасывается и цикл повторяется. При равенстве содержимого счетного регистра и регистра OCR1A состояние вывода PB1 (OC1A) изменяется в соответствии со значениями разрядов COM1A1 и COM1A0 регистра управления TCCR1 (табл. 1.9). Выбор источника

тактового сигнала задается разрядами CS13...CS10 этого регистра (табл. 1.10), а включение режима ШИМ — записью 1 в разряд PWM1 (табл. 1.8).

Таблица 1.8 — Формат регистра TCCR1

7	6	5	4	3	2	1	0
CTC1	PWM1	COM1A1	COM1A0	CS13	CS12	CS11	CS10

Таблица 1.9 — Поведение вывода PB1 (OC1A) в режиме ШИМ

COM1A1	COM1A0	PB1 (OC1A)
0	0	Таймер/счетчик T1 отключен от вывода
0	1	Таймер/счетчик T1 отключен от вывода
1	0	Сбрасывается в 0 при равенстве TCNT1 и OCR1A, устанавливается в 1 при TCNT1=$00
1	1	Устанавливается в 1 при равенстве TCNT1 и OCR1A, сбрасывается в 0 при TCNT1=$00

Таблица 1.10 — Выбор источника тактового сигнала для таймера T1

Регистр TCCR1				Источник тактового сигнала
CS13	CS12	CS11	CS10	
0	0	0	0	Таймер/счетчик остановлен
0	0	0	1	CKx16
0	0	1	0	CKx8
0	0	1	1	CKx4
0	1	0	0	CKx2
0	1	0	1	CK (тактовый сигнал МК)
0	1	1	0	CK/2
0	1	1	1	CK/4
1	0	0	0	CK/8
1	0	0	1	CK/16
1	0	1	0	CK/32
1	0	1	1	CK/64
1	1	0	0	CK/128
1	1	0	1	CK/256
1	1	1	0	CK/512
1	1	1	1	CK/1024

Содержимое регистра сравнения OCR1B определяет частоту ШИМ-сигнала (табл. 1.11), содержимое регистра сравнения OCR1A определяет скважность ШИМ-сигнала.

Таблица 1.11 — Зависимость частоты ШИМ-сигнала от тактовой частоты

Частота тактового сигнала таймера/счетчика	Содержимое регистра OCR1B	Частота ШИМ-сигнала (кГц)
СК	159	10
СКх2	159	20
СКх4	213	30
СКх4	159	40
СКх8	255	50
СКх8	213	60
СКх8	181	70
СКх8	159	80
СКх8	141	90
СКх16	255	100
СКх16	231	110
СКх16	213	120
СКх16	195	130
СКх16	181	140
СКх16	169	150

При работе таймера/счетчика T1 в режиме ШИМ может генерироваться прерывание по переполнению T1, а также прерывание от схемы сравнения (см. флаги и биты разрешения регистров TIFR и TIMSK).

В микроконтроллере ATtiny15L имеется встроенный синтезатор частоты, формирующий сигнал с частотой, в 16 раз превышающей частоту тактового сигнала встроенного RC-генератора. Номинальная частота RC-генератора равна 1,6 МГц, а частота на выходе синтезатора частоты равна 25,6 МГц.

Основная функция сторожевого таймера — защита устройства от сбоев. Благодаря сторожевому таймеру можно прервать выполнение

зациклившейся программы. Если сторожевой таймер включен, то через определенные промежутки времени выполняется сброс микроконтроллера. При нормальном выполнении программы сторожевой таймер должен периодически сбрасываться командой WDR.

Для управления сторожевым таймером предназначен регистр WDTCR (табл. 1.12). Краткое описание разрядов этого регистра приведено в табл. 1.13. Непосредственно перед включением сторожевого таймера рекомендуется выполнить его сброс командой WDR.

Таблица 1.12 — Формат регистра WDTCR

7	6	5	4	3	2	1	0
–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0

Таблица 1.13 — Разряды регистра WDTCR

Разряд	Название	Описание
7...5	–	Зарезервированы
4	WDTOE	Разрешение выключения сторожевого таймера
3	WDE	Разрешение включения сторожевого таймера
2...0	WDP2...WDP0	Коэффициент деления предделителя частоты

Период наступления тайм-аута сторожевого таймера задается с помощью разрядов WDP2...WDP0 согласно табл. 1.14. Сторожевой таймер имеет независимый тактовый генератор с номинальным значением частоты 1 МГц и может работать даже в режиме PowerDown.

Таблица 1.14 — Задание периода сторожевого таймера

WDP2	WDP1	WDP0	Число тактов генератора
0	0	0	16K
0	0	1	32K
0	1	0	64K
0	1	1	128K
1	0	0	256K
1	0	1	512K
1	1	0	1024K
1	1	1	2048K

Стек

В микроконтроллерах AVR семейства Tiny стек реализован аппаратно. Глубина стека равна трем уровням, а разрядность равна размеру счетчика команд (9 разрядов). При вызове подпрограммы адрес команды, расположенной за командой RCALL, сохраняется в стеке. При возврате из подпрограммы этот адрес извлекается из стека и загружается в счетчик команд. То же происходит и во время прерывания программы.

Непосредственно из программы стек недоступен, так как в наборе команд микроконтроллера отсутствуют команды занесения в стек и извлечения из стека. Указатель стека также недоступен из программы. Микроконтроллер сам управляет перемещением данных по стеку.

Энергонезависимая память данных EEPROM

Для обращения к EEPROM (ее объем составляет 64 байта) используются три регистра ввода/вывода: регистр адреса EEAR, регистр данных EEDR и регистр управления EECR (табл. 1.15 и 1.16).

Таблица 1.15 — Формат регистра EECR

7	6	5	4	3	2	1	0
–	–	–	–	EERIE	EEMWE	EEWE	EERE

Таблица 1.16 — Разряды регистра EECR

Разряд	Название	Описание
7...4	–	Не используются, читаются как 0
3	EERIE	Разрешение прерывания от EEPROM. Данный разряд управляет генерацией прерывания, возникающего при завершении цикла записи в EEPROM. Если этот разряд установлен в 1, прерывания разрешены (если флаг I регистра SREG также установлен в 1). При сброшенном разряде

		EEMWEпрерывание генерируется постоянно
2	EEMWE	Управление разрешением записи в EEPROM. После программной установки этот разряд сбрасывается аппаратно через 4 такта
1	EEWE	Разрешение записи в EEPROM. При установке этого разряда в 1 происходит запись данных в EEPROM, если EEMWE=1
0	EERE	Разрешение чтения из EEPROM. По окончании чтения сбрасывается аппаратно

Для записи одного байта в EEPROM необходимо:

дождаться готовности EEPROM к записи (ждать, пока не сбросится флаг EEWE);

загрузить байт данных в регистр EEDR, а требуемый адрес — в регистр EEAR;

установить в 1 флаг EEMWE;

в течение 4-х машинных циклов после установки EEMWE записать 1 в разряд EEWE.

Рекомендуется запрещать все прерывания при выполнении пунктов 2...4 описанной последовательности. Длительность процесса записи составляет 4...8 мс. Процедура чтения из EEPROM гораздо проще. После загрузки требуемого адреса в регистр EEAR программа должна установить в 1 разряд EERE. Когда запрошенные данные будут находиться в регистре данных EEDR, произойдет аппаратный сброс этого разряда.

Аналоговый компаратор

Будучи включенным, компаратор позволяет сравнить значения напряжений на выводах PB0 и PB1. Чтобы указанные линии порта могли использоваться аналоговым компаратором, они должны быть сконфигурированы как входы. Внутренние подтягивающие резисторы, если

они подключены, при разрешении работы компаратора отключаются автоматически.

Результатом сравнения является логическое значение, которое может быть прочитано из программы. По результату сравнения может быть сгенерировано прерывание. Управление работой компаратора осуществляется с помощью битов регистра ACSR (табл. 1.17—1.18). При включении напряжения питания все разряды регистра ACSR сбрасываются в 0. К не инвертирующему входу компаратора вместо вывода AIN0 микроконтроллера может быть подключен внутренний источник опорного напряжения величиной 1.22±0.05 В.

Таблица 1.17 — Разряды регистра ACSR

Разряд	Название	Описание
7	ACD	Выключение компаратора (1 — выключен)
6	ACBG	Подключение к неинвертирующему входу компаратора внутреннего ИОН (1 — подключен, 0 — не подключен)
5	ACO	Результат сравнения (выход компаратора)
4	ACI	Флаг прерывания от компаратора
3	ACIE	Разрешение прерывания от компаратора
2	–	Зарезервирован
1,0	ACIS1:ACIS0	Условия возникновения прерывания от компаратора

Таблица 1.18 — Условия генерации запроса на прерывание от компаратора

ACIS1	ACIS0	Условие
0	0	Любое изменение состояния выхода компаратора
0	1	Зарезервировано
1	0	Изменение состояния выхода компаратора с 1 на 0
1	1	Изменение состояния выхода компаратора с 0 на 1

Аналого-цифровой преобразователь

В процессе работы АЦП может функционировать в двух режимах:

режим одиночного преобразования — запуск каждого преобразования инициируется пользователем;

режим непрерывного преобразования — запуск преобразований выполняется непрерывно через определенные интервалы времени.

Управление модулем АЦП и контроль его состояния осуществляется с помощью регистра ADCSR (табл. 1.19).

Таблица 1.19 — Разряды регистра ADCSR

Разряд	Название	Описание
7	ADEN	Разрешение АЦП (1 — включен)
6	ADSC	Запуск преобразования (1 — начать преобразование)
5	ADFR	Выбор режима работы АЦП (0 — одиночное преобразование)
4	ADIF	Флаг прерывания от АЦП
3	ADIE	Разрешение прерывания от АЦП
2...0	ADPS2:ADPS0	Выбор частоты преобразования

Таблица 1.20 — Задание коэффициента деления предделителя АЦП

ADRS2	ADRS1	ADRS0	Коэффициент деления
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Наибольшая точность преобразования достигается при тактовой частоте модуля АЦП в диапазоне 50...200 кГц. Для этого тактовая частота

микроконтроллера поступает на АЦП через предделитель с программируемым коэффициентом деления. Для повышения точности преобразования (чтобы свести к минимуму помехи, наводимые ядром процессора) в микроконтроллере предусмотрен специальный спящий режим — ADCNoiseReduction. В этом режиме из всех периферийных устройств функционируют только АЦП и сторожевой таймер. Сразу же после остановки процессора начнется цикл преобразования. По завершении преобразования будет сгенерировано прерывание от АЦП, которое переведет микроконтроллер в рабочий режим, и начнется выполнение подпрограммы обработки этого прерывания.

Поскольку АЦП 10-разрядный, результат преобразования размещен в двух регистрах, доступных только для чтения: ADCH и ADCL. Сначала необходимо прочитать ADCL, а затем ADCH. Если достаточно точности восьми разрядов, для получения результата достаточно прочитать содержимое регистра ADCH.

Номер активного канала (аналоговый вход, подключаемый ко входу АЦП) и источника опорного напряжения задается регистром ADMUX (табл. 1.21—1.23). Разряд ADLAR служит для управления выравниванием результата преобразования. Если этот разряд установлен в 1, результат преобразования выравнивается по левой границе 16-разрядного слова, если сброшен в 0 — по правой границе.

Таблица 1.21 — Разряды регистра ADMUX

Разряд	Название	Описание
7,6	REFS1:REFS0	Выбор источника опорного напряжения
5	ADLAR	Выравнивание результата преобразования
4,3	–	Зарезервировано
2...0	MUX2:MUX0	Выбор входного канала

Таблица 1.22 — Выбор источника опорного напряжени

REFS1	REFS0	Источник опорного напряжения
0	0	Напряжение питания микроконтролера
0	1	Внешний ИОН, подключенный к выводу PB0, внутренний ИОН отключен
1	0	Внутренний ИОН напряжением 2,56 В, отключенный от вывода PB0 (AREF)
1	1	Внутренний ИОН напряжением 2,56 В, подключенный к выводу PB0 (AREF)

Таблица 1.23 — Номер активного канал

MUX2	MUX 1	MUX 0	Вход
0	0	0	ADC0 (PB5)
0	0	1	ADC1 (PB2)
0	1	0	ADC2 (PB3)
0	1	1	ADC3 (PB4)

Программа работы

1. Загрузить для отладки в AVRStudio программу преобразования целых 16-битных чисел в двоично-десятичные числа. Алгоритм программы «**bin16BCD5**» заключается в следующем. Предположим, что имеется целое беззнаковое 16-битное число (диапазон от 0 до 65535). Очевидно, что необходимо найти 5 десятичных цифр. Способ преобразования заключается в том, чтобы, вычитая из исходного числа число 10000, сначала определить десятичную цифру десятков тысяч. Затем находится цифра тысяч последовательным вычитанием числа 1000 и т.д. Вычитание каждый раз производится до получения отрицательной разности с подсчетом числа вычитаний. При переходе к определению каждого следующего десятичного разряда в регистрах исходного числа восстанавливается последняя положительная разность. После того, как будет найдена десятичная цифра десятков, в регистрах исходного числа останется десятичная цифра единиц.

Проследить выполнение программы в пошаговом и автоматическом режиме, записав предварительно в регистры r16 и r17 шестнадцатеричное

число $NNNN, где N — номер варианта, рассчитанный по методике ТМЦ ДО (число от 1 до 9). Вокне I/O раскройтесодержимое Register 1—31, Processor, I/O ATTINY15 (CPU, WATCHDOG). Какие команды программы влияют на флаги регистра статуса SREG? Зафиксируйте в отчете результат преобразования. В программе часто используются команды вычитания константы из регистра. Есть ли в системе команд AVR аналогичные команды сложения регистра и константы? Как будет работать программа, если в ней удалить последнюю команду?

```
;***** Программа bin16BCD5
.DEVICEATtiny15        ; Определить устройство
.INCLUDE
"C:\Program Files\Atmel\AVR Tools\AvrAssembler\Appnotes\tn15def.inc"
; Вложить файл определения адресов регистров ввода\вывода

;***** Регистровые переменные
.deffbinL    =r16        ; двоичное значение, младший байт
.deffbinH    =r17        ; двоичное значение, старший байт
.def   tBCD0  =r17         ; BCD значение, цифры 1 и 0
.def   tBCD1  =r18         ; BCD значение, цифры 3 и 2
.def   tBCD2  =r19         ; BCD значение, цифра 4
; Переменные fbinH и tBCD0 должны размещаться в одном регистре
WDR                        ; Сброс сторожевого таймера
ldir20,0b00001000          ; Включение сторожевого
outWDTCR,r20               ; таймера
lditBCD2, -1               ; Начало преобразования
m1:
inc    tBCD2
subifbinL, low(10000)
sbcifbinH, high(10000)
brsh   m1
subifbinL, low(-10000)
sbcifbinH, high(-10000)
ldi    tBCD1, -0x11
m2:
subi   tBCD1, -0x10
subifbinL, low(1000)
sbcifbinH, high(1000)
brsh   m2
subifbinL, low(-1000)
```

```
sbcifbinH, high(-1000)
m3:
inc    tBCD1
subifbinL, low(100)
sbcifbinH, high(100)
brsh   m3
subifbinL, -100
ldi    tBCD0, -0x10
m4:
subi   tBCD0, -0x10
subifbinL, 10
brsh   m4
subifbinL, -10
add    tBCD0, fbinL     ; Конец преобразования

m5:  rjmpm5                    ; Зацикливание программы
```

Какой период срабатывания сторожевого таймера задан в программе? Что будет, если дождаться его срабатывания?

2. Загрузить для отладки в AVRStudio программу**CLOK**,реализующую двоично-десятичный счетчик на регистре r19. Счетчик считает с частотой прерываний по переполнению таймера T0. Тактовый сигнал на вход таймера подается через программируемый делитель частоты. Коэффициент пересчета счетчика равен 100. Для счета используются вспомогательные регистры r16 (счет единиц), r17 (счет десятков) и r18 (объединение десятков и единиц). Основная программа обнуляет регистры счетчика, устанавливает режим работы T0, разрешает прерывания по переполнению T0 и зацикливается. Двоично-десятичный счет реализуется в подпрограмме прерывания, расположенной начиная с адреса вектора прерывания по переполнению таймера T0. Заметим, что в системе команд AVR нет команды сложения регистра с константой и команды десятичной коррекции аккумулятора, как и самого аккумулятора.

Набрать исходный текст программы CLOK.asm без комментария. Проверить работу в пошаговом и автоматическом режимах. Вокне I/O AVR Studio раскройтесодержимое Register 1—31, I/O ATTINY15 (CPU, TIMER_COUNTER_0).

```
;******Программа CLOK
.DEVICE ATtiny15
.INCLUDE
 "C:\Program Files\Atmel\AVR Tools\AvrAssembler\Appnotes\tn15def.inc"

      rjmp  RESET
.org  $005                  ; Вектор прерывания по переполнению T0
      inc   r16             ; Инкремент единиц
      cpi   r16,$0A
      breq  m1
      rjmp  m3
m1:   clr   r16
      subi  r17,-$10        ; Инкремент десятков
      cpi   r17,$A0
      breq  m2
      rjmp  m3
m2:   clr   r17
m3:   mov   r18,r17
      add   r18,r16
      mov   r19,r18         ; Инкремент двоично-десятичного счета
      reti

RESET:
      clr   r16             ; Обнуление регистров
      clr   r17
      clr   r18
      clr   r19
      ldi   r20,0b00000001  ; Выбор источника тактового сигнала для T0
      out   TCCR0,r20
      ldi   r20,0b00000010  ; Разрешение прерываний
out   TIMSK,r20       ; по переполнению таймера T0
      sei                   ; Глобальное разрешение прерываний
m4:   rjmp  m4
```

С какой частотой переполняется T0? Каким образом можно уменьшить скорость счета в 1024 раза? Объяснить поведение регистров SREG и TIFR

при работе программы. Пояснить назначение директивы .DEVICE. Пояснить содержимое файла clok.map в окне Project.

Изменить программу так, чтобы уменьшить коэффициент пересчета счетчика до 10N, где N — номер варианта. В отчете представить измененный вариант программы с комментарием.

3. Сформировать на выводе PB1 (OC1A) микроконтроллера ШИМ-сигнал с частотой 50 кГц (программа **PWM1**). Таймер T1 используется как генератор импульсов с программируемым периодом (содержимое регистра сравнения OCR1B) и длительностью (содержимое регистра сравнения OCR1A). В окне I/O раскройте содержимое Register 1—31, Processor, I/OATTINY15 (PORTB, TIMER_COUNTER_1).

```
;****** ПрограммаPWM1
.DEVICE ATtiny15
.INCLUDE
"C:\Program Files\Atmel\AVR Tools\AvrAssembler\Appnotes\tn15def.inc"

      sbi   DDRB,1          ; Настройка первой линии порта B на вывод
      ldi   r16,0b01100010  ; Режим работы T1 (ШИМ с частотой
                            ; тактирования 12.8 МГц, 1 при сбросе,
; 0 при сравнении)
      out   TCCR1,r16
      ldi   r16,0xFF        ; Частота импульсов 50 кГц
      out   OCR1B,r16
      ldi   r16,0x80        ; Скважность импульсов примерно 2
      out   OCR1A,r16
m1:   rjmp  m1
```

Проследить работу программы в пошаговом режиме. На сколько меняется содержимое T1 при выполнении команды rjmpm1? Почему? Какие флаги устанавливаются в регистре TIFR?

Модифицировать программу так, чтобы частота ШИМ составила 20 кГц, а скважность — 4 (отношение периода к длительности импульса).

4. Проверить программу обращения к энергонезависимой памяти данных EEPROM (проект **EEPROM**). Открыть окна для просмотра регистров общего назначения 16—31, регистров ввода/вывода (CPU, EEPROM), памяти EEPROM (Memory 3). Проследите выполнение программы в пошаговом режиме. Когда выполняется подпрограмма прерывания и что она делает? Специального флага прерываний от EEPROM нет, поэтому при обращении к подпрограмме прерывания флаг не сбрасывается и прерывания генерируются постоянно.

```
;****** Программа EEPROM
.DEVICE ATtiny15
.INCLUDE
"C:\Program Files\Atmel\AVR Tools\AvrAssembler\Appnotes\tn15def.inc"

.cseg
      ; Рабочие переменные
      .def   AddrReg=r20
      .def   Data1Reg=r21
      .def   Data2Reg=r22

      ; Векторыпрерываний
      rjmp  RESET
      reti
      reti
      reti
      reti
      reti
      rjmp  EEPROM_READY  ; ВекторпрерыванияпозаписивEEPROM
      reti
      reti
EEPROM_READY:             ; Подпрограмма прерывания по окончании
      inc    r25          ; цикла записи в EEPROM
      reti
EEWrite:                  ; Подпрограмма записи байта в EEPROM
      sbic   EECR,EEWE    ; Ждать, пока флаг EEWE
      rjmp   EEWRite      ; не будет сброшен
      cli                 ; Запретить прерывания
      out    EEAR,AddrReg ; Загрузить адрес
      out    EEDR,Data1Reg ; Загрузить данные
      sbi    EECR,EEMWE
```

```
        sbi     EECR,EEWE       ; Выдать строб записи байта в EEPROM
        sbi     EECR,EERIE      ; Разрешить прерывание по завершении
        sei                     ; цикла записи в EEPROM
        cbi     EECR,EERIE      ; Запретить дальнейшие прерывания
        ret
EERead:                         ; Подпрограмма чтения байта EEPROM
        sbic    EECR,EEWE       ; Ждать окончания текущей записи, пока
        rjmp    EERead          ; флаг EEWE не равен 0
        rjmp    EEWRite
        out     EEAR,AddrReg    ; Загрузитьадрес
        sbi     EECR,EERE       ; Выдать строб чтения из EEPROM
        in      Data2Reg,EEDR   ; Прочитанный байт в регистр
        ret
RESET:                          ; Начало основной программы
        clr     r25             ; Очистка регистров
        clr     r21
        clr     r22
        ldi     AddrReg,$18
        ldi     Data1Reg,$DD
        rcall   EEWrite         ; Вызовподпрограммызаписив EEPROM
        rcall   EERead          ; Вызов подпрограммы чтения из EEPROM
m1:     rjmp    m1
```

Изменить программу так, чтобы она записывала в ячейку NEEPROM число 100+N, а читала записанный байт в регистр rN, где N — номер варианта (число от 1 до 9).

Контрольные вопросы

1. Где сохраняется адрес возврата при обращении к подпрограммам? Почему в программной модели микроконтроллера ATtiny15 нет указателя стека? Какую разрядность имеет стек ATtiny15 и сколько в нем ячеек? Можно ли реализовать в ATtiny15 вложенные прерывания программы?

2. Перечислите все источники прерываний в ATtiny15 в порядке убывания приоритета.

3. Каким образом программируется FLASH-память программ ATtiny15?

4. Какой командой микроконтроллер переводится в «спящий» режим?

5. Биты каких регистров можно устанавливать и сбрасывать командами sbi и cbi?

Содержание отчета

Отчет в формате WORD должен содержать тексты измененных (в соответствии с вариантом задания) программ с комментариями, ответы на вопросы по пунктам работы, рисунки, отображающие окна регистров и памяти, ответы на контрольные вопросы.

Перечень команд микроконтроллера ATtiny15L

Арифметические и логические команды

ADD	Rd, Rr	Add Two Registers
ADC	Rd, Rr	Add with Carry Two Registers
SUB	Rd, Rr	Subtract Two Registers
SUBI	Rd, K	Subtract Constant from Register
SBC	Rd, Rr	Subtract with Carry Two Registers
SBCI	Rd, K	Subtract with Carry Constant from Reg.
AND	Rd, Rr	Logical AND Registers
ANDI	Rd, K	Logical AND Register and Constant
OR	Rd, Rr	Logical OR Registers
ORI	Rd, K	Logical OR Register and Constant
EOR	Rd, Rr	Exclusive OR Registers
COM	Rd	One's Complement
NEG	Rd	Two's Complement

SBR	Rd, K	Set Bit(s) in Register
CBR	Rd, K	Clear Bit(s) in Register
INC	Rd	Increment
DEC	Rd	Decrement
TST	Rd	Test for Zero or Minus
CLR	Rd	Clear Register
SER	Rd	Set Register

Команды передачи управления

RJMP	k	Relative Jump
RCALL	k	Relative Subroutine Call
RET		Subroutine Return
RETI		Interrupt Return
CPSE	Rd, Rr	Compare, Skip if Equal)
CP	Rd, Rr	Compare
CPC	Rd, Rr	Compare with Carry
CPI	Rd, K	Compare Register with Immediate
SBRC	Rr, b	Skip if Bit in Register Cleared
SBRS	Rr, b	Skip if Bit in Register is Set
SBIC	P, b	Skip if Bit in I/O Register Cleared)
SBIS	P, b	Skip if Bit in I/O Register is Set
BRBS	s, k	Branch if Status Flag Set
BRBC	s, k	Branch if Status Flag Cleared
BREQ	k	Branch if Equal
BRNE	k	Branch if Not Equal
BRCS	k	Branch if Carry Set
BRCC	k	Branch if Carry Cleared
BRSH	k	Branch if Same or Higher
BRLO	k	Branch if Lower
BRMI	k	Branch if Minus

BRPL	k	Branch if Plus
BRGE	k	Branch if Greater or Equal, Signed
BRLT	k	Branch if Less Than Zero, Signed
BRHS	k	Branch if Half-carry Flag Set
BRHC	k	Branch if Half-carry Flag Cleared
BRTS	k	Branch if T-flag Set
BRTC	k	Branch if T-flag Cleared
BRVS	k	Branch if Overflow Flag is Set
BRVC	k	Branch if Overflow Flag is Cleared
BRIE	k	Branch if Interrupt Enabled
BRID	k	Branch if Interrupt Disabled

Команды пересылки данных

LD	Rd, Z	Load Register Indirect
ST	Z, Rr	Store Register Indirect
MOV	Rd, Rr	Move between Registers
LDI	Rd, K	Load Immediate
IN	Rd, P	In Port
OUT	P, Rr	Out Port
LPM		Load Program Memory

Команды операций с битами

SBI	P, b	Set Bit in I/O Register
CBI	P, b	Clear Bit in I/O Register
LSL	Rd	Logical Shift Left
LSR	Rd	Logical Shift Right
ROL	Rd	Rotate Left through Carry
ROR	Rd	Rotate Right through Carry
ASR	Rd	Arithmetic Shift Right
SWAP	Rd	Swap Nibbles

BSET	s	Flag Set
BCLR	s	Flag Clear
BST	Rr, b	Bit Store from Register to T
BLD	Rd, b	Bit Load from T to Register
SEC		Set Carry
CLC		Clear Carry
SEN		Set Negative Flag
CLN		Clear Negative Flag
SEZ		Set Zero Flag
CLZ		Clear Zero Flag
SEI		Global Interrupt Enable
CLI		Global Interrupt Disable
SES		Set Signed Test Flag
CLS		Clear Signed Test Flag
SEV		Set Two's Complement Overflow
CLV		Clear Two's Complement Overflow
SET		Set T in SREG
CLT		Clear T in SREG
SEH		Set Half-carry Flag in SREG
CLH		Clear Half-carry Flag in SREG
NOP		No Operation
SLEEP		Sleep
WDR		Watchdog Reset

Лабораторная работа № 2 Микроконтроллер ATMEGA8

Цель работы. Целью лабораторной работы является отладка прикладных программ для микроконтроллера AVR семейства Mega с помощью персонального компьютера и программных средств отладки.

Отличительные особенности ATmega8

8-разрядный высокопроизводительный AVR-микроконтроллер с малым потреблением.

Прогрессивная RISC-архитектура:

- 130 высокопроизводительных команд, большинство команд выполняется за один тактовый цикл.
- 32 8-разрядных рабочих регистра общего назначения.

Полностью статическая работа.

Приближающаяся к 16 MIPS (при тактовой частоте 16 МГц) производительность.

Встроенный 2-цикловый перемножитель.

Энергонезависимая память программ и данных:

- 8 Кбайт внутрисистемнопрограммируемойFlash-памяти.
- Обеспечивает 1000 циклов стирания/записи.
- Дополнительный сектор загрузочных кодов с независимыми битами блокировки.
- 512 байт EEPROM.
- Обеспечивает 100000 циклов стирания/записи.
- 1 Кбайт встроенной SRAM.
- Программируемая блокировка, обеспечивающая защиту программных средств пользователя.

Встроенная периферия:

Два 8-разрядных таймера/счетчика с отдельным предварительным делителем, один с режимом сравнения.

Один 16-разрядный таймер/счетчик с отдельным предварительным делителем и режимами захвата и сравнения.

Счетчик реального времени с отдельным генератором.

Три канала ШИМ (PWM).

8-канальный аналого-цифровой преобразователь (в корпусах TQFP и MLF).

6 каналов с 10-разрядной точностью.

2 канала с 8-разрядной точностью.

6-канальный аналого-цифровой преобразователь (в корпусе PDIP).

4 канала с 10-разрядной точностью.

2 канала с 8-разрядной точностью.

Байт-ориентированный 2-проводный последовательный интерфейс.

Программируемый последовательный USART.

Последовательный интерфейс SPI (ведущий/ведомый).

Программируемый сторожевой таймер с отдельным встроенным генератором.

Встроенный аналоговый компаратор.

Специальные микроконтроллерные функции:

Сброс по подаче питания и программируемый детектор кратковременного снижения напряжения питания.

Встроенный калиброванный RC-генератор.

Внутренние и внешние источники прерываний.

Пять режимов пониженного потребления: Idle, Power-save, Power-down, Standby и снижения шумов ADC.

Выводы I/O и корпуса:

23 программируемые линии ввода/вывода.

28-выводной корпус PDIP, 32-выводной корпус TQFP (рис. 2.1).

Рабочие напряжения:

2,7 — 5,5 В (ATmega8L).

4,5 — 5,5 В (ATmega8).

Рабочая частота:

0 — 8 МГц (ATmega8L).

0 — 16 МГц (ATmega8).

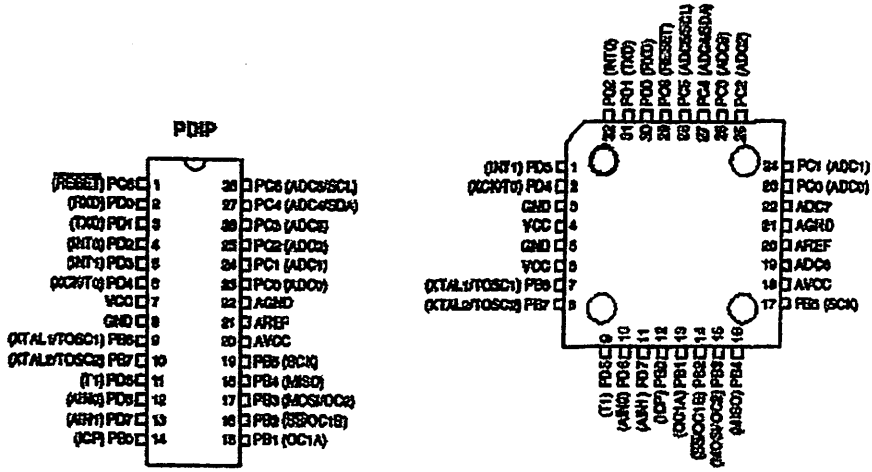


Рис. 2.1 — Расположение выводов ATmega8

Таблица 2.1 — Описание выводов ATmega8

Обозначение	Описание
PB0 (ICP)	B0 (Вход захвата таймера/счетчика T1)
PB1 (OC1A)	B1 (Выход сравнения A таймера/счетчика T1)
PB2 (SS/OC1B)	B2 (Выбор Slave-устройства в канале SPI/ выход сравнения B таймера/счетчика T1)
PB3 (MOSI/OC2)	B3 (Выход (Master) или вход (Slave) данных канала SPI/выход сравнения таймера/счетчика T2)
PB4 (MISO)	B4 (Вход (Master) или выход (Slave) данных канала SPI
PB5 (SCK)	B5 (Выход (Master) или вход (Slave) тактового сигнала SPI
PB6 (XTAL1/TOSC1)	B6 (Вход тактового генератора/вывод для подключения резонатора к таймеру/счетчику T2)
PB7 (XTAL2/TOSC2)	B7 (Выход тактового генератора/вывод для подключения резонатора к таймеру/счетчику T2)
PC0 (ADC0)	C0 (Вход АЦП)
PC1 (ADC1)	C1 (Вход АЦП)

Таблица 2.1 — Описание выводов ATmega8

Обозначение	Описание
PC2 (ADC2)	C2 (Вход АЦП)
PC3 (ADC3)	C3 (Вход АЦП)
PC4 (ADC4/SDA)	C4 (Вход АЦП/линия данных модуля TWI)
PC5 (ADC5/SCL)	C5 (Вход АЦП/тактовый сигнал модуля TWI)
PC6 (RESET)	C6 (Вход сброса)
ADC6	Вход АЦП
ADC7	Вход АЦП
PD0 (RXD)	D0 (Вход USART)
PD1 (TXD)	D1 (Выход USART)
PD2 (INT0)	D2 (Вход внешнего прерывания)
PD3 (INT1)	D3 (Вход внешнего прерывания)
PD4 (T0/XCK)	D4 (Вход внешнего тактового сигнала таймера/счетчика T0/тактовый сигнал USART)
PD5 (T1)	D5 (Вход внешнего тактового сигнала таймера/счетчика T1)
PD6 (AIN0)	D6 (Неинвертирующий вход компаратора)
PD7 (AIN1)	D7 (Инвертирующий вход компаратора)
AREF	Вход опорного напряжения для АЦП
AGND	Аналоговый общий вывод
AVcc	Вывод источника питания АЦП
GND	Общий вывод
Vcc	Вывод источника питания

Таблица 2.2 — Регистры ввода/вывода (адрес, название и функция)

0x3F (0x5F)	SREG	Регистр состояния
0x3E (0x5E)	SPH	Указатель стека, старший байт
0x3D (0x5D)	SPL	Указатель стека, младший байт
0x3B (0x5B)	GICR	Общий регистр управления прерываниями
0x3A (0x5A)	GIFR	Общий регистр флагов прерываний
0x39 (0x59)	TIMSK	Маски прерываний от таймеров/счетчиков
0x38 (0x58)	TIFR	Флаги прерываний от таймеров/счетчиков
0x37 (0x57)	SPMCR	Регистр управления памятью программ
0x36 (0x56)	TWCR	Регистр управления TWI
0x35 (0x55)	MCUCR	Регистр управления микроконтроллером

0x34 (0x54)	MCUCSR	Регистр управления и состояния МК
0x33 (0x53)	TCCR0	Регистр управления таймером/счетчиком T0
0x32 (0x52)	TCNT0	Счетный регистр таймера/счетчика T0
0x31 (0x51)	OSCCAL	Регистр калибровки тактового генератора
0x30 (0x50)	SFIOR	Регистр специальный функций
0x2F (0x4F)	TCCR1A	Регистр управления A таймера/счетчика T1
0x2E (0x4E)	TCCR1B	Регистр управления B таймера/счетчика T1
0x2D (0x4D)	TCNT1H	Счетный регистр T1, старший байт
0x2C (0x4C)	TCNT1L	Счетный регистр T1, младший байт
0x2B (0x4B)	OCR1AH	Регистр совпадения A T1, старший байт
0x2A (0x4A)	OCR1AL	Регистр совпадения A T1, младший байт
0x29 (0x49)	OCR1BH	Регистр совпадения B T1, старший байт
0x28 (0x48)	OCR1BL	Регистр совпадения B T1, младший байт
0x27 (0x47)	ICR1H	Регистр захвата T1, старший байт
0x26 (0x46)	ICR1L	Регистр захвата T1, младший байт
0x25 (0x45)	TCCR2	Регистр управления таймера/счетчика T2
0x24 (0x44)	TCNT2	Счетный регистр таймера/счетчика T2
0x23 (0x43)	OCR2	Регистр совпадения таймера/счетчика T2
0x22 (0x42)	ASSR	Регистр состояния асинхронного режима
0x21 (0x41)	WDTCR	Регистр управления сторожевым таймером
0x20 (0x40)	UBRRH	Регистр управления USART
0x1F (0x3F)	EEARH	Регистр адреса EEPROM, старший байт
0x1E (0x3E)	EEARL	Регистр адреса EEPROM, младший байт
0x1D (0x3D)	EEDR	Регистр данных EEPROM
0x1C (0x3C)	EECR	Регистр управления EEPROM
0x18 (0x38)	PORTB	Регистр данных порта B
0x17 (0x37)	DDRB	Регистр направления данных порта B

0x16 (0x36)	PINB	Выводы порта B
0x15 (0x35)	PORTC	Регистр данных порта C
0x14 (0x34)	DDRC	Регистр направления данных порта C
0x13 (0x33)	PINC	Выводы порта C
0x12 (0x32)	PORTD	Регистр данных порта D
0x11 (0x31)	DDRD	Регистр направления данных порта D
0x10 (0x30)	PIND	Выводы порта D
0x0F (0x2F)	SPDR	Регистр данных SPI
0x0E (0x2E)	SPSR	Регистр состояния SPI
0x0D (0x2D)	SPCR	Регистр управления SPI
0x0C (0x2C)	UDR	Регистр данных USART
0x0B (0x2B)	UCSRA	Регистр управления и состояния AUSART
0x0A (0x2A)	UCSRB	Регистр управления и состояния BUSART
0x09 (0x29)	UBRRL	Регистр скорости передачи USART
0x08 (0x28)	ACSR	Состояние аналогового компаратора
0x07 (0x27)	ADMUX	Регистр управления мультиплексором АЦП
0x06 (0x26)	ADCSR	Регистр управления и состояния АЦП
0x05 (0x25)	ADCH	Регистр данных АЦП, старший байт
0x04 (0x24)	ADCL	Регистр данных АЦП, младший байт
0x03 (0x23)	TWDR	Регистр данных TWI
0x02 (0x22)	TWAR	Регистр адреса TWI
0x01 (0x21)	TWSR	Регистр состояния TWI
0x00 (0x20)	TWBR	Регистр скорости передачи TWI

Таблица 2.3 — Векторы прерываний (номер, адрес, источник и описание)

1	0x000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset
2	0x001	INT0	External Interrupt Request 0
3	0x002	INT1	External Interrupt Request 1
4	0x003	TIMER2 COMP	Timer/Counter2 Compare Match
5	0x004	TIMER2 OVF	Timer/Counter2 Overflow
6	0x005	TIMER1 CAPT	Timer/Counter1 Capture Event
7	0x006	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	0x007	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	0x008	TIMER1 OVF	Timer/Counter1 Overflow
10	0x009	TIMER0 OVF	Timer/Counter0 Overflow
11	0x00A	SPI, STC	Serial Transfer Complete
12	0x00B	USART, RXC	USART, Rx Complete
13	0x00C	USART, UDRE	USART Data Register Empty
14	0x00D	USART, TXC	USART, Tx Complete
15	0x00E	ADC	ADC Conversion Complete
16	0x00F	EE_RDY	EEPROM Ready
17	0x010	ANA_COMP	Analog Comparator
18	0x011	TWI	Two-wire Serial Interface
19	0x012	SPM_RDY	Store Program Memory Ready

Порты ввода-вывода

Все порты ввода-вывода (ПВВ) AVR-микроконтроллеров работают по принципу чтение-модификация-запись при использовании их в качестве портов универсального ввода-вывода. Это означает, что изменение направления ввода-вывода одной линии порта командами SBI и CBI будет происходить без ложных изменений направления ввода-вывода других линий порта. Данное распространяется также и на изменение логического уровня (если линия порта настроена на вывод) или на включение/отключение подтягивающих резисторов (если линия настроена на ввод). Каждый выходной буфер имеет симметричную характеристику управления с высоким втекающим и вытекающим выходными токами. Выходной драйвер обладает нагрузочной способностью, которая позволяет непосредственно управлять светодиодными индикаторами. Ко всем линиям портов может быть подключен индивидуальный выборочный подтягивающий к плюсу питания резистор, сопротивление которого не зависит от напряжения питания. Ссылки на регистры и биты регистров в данном разделе даны в общей форме. При этом символ «x» заменяет наименование ПВВ, а символ «n» заменяет номер разряда ПВВ. Однако при составлении программы необходимо использовать точную форму записи. Например, PORTB3, означающий разряд 3 порта B, в данном случае записывается как PORTxn.

Для каждого порта ввода-вывода в памяти ввода-вывода зарезервировано три ячейки: одна под регистр данных — PORTx, другая под регистр направления данных — DDRx и третья под состояние входов порта — PINx. Ячейка, хранящая состояние на входах портов, доступна только для чтения, а регистры данных и направления данных имеют двунаправленный доступ. Кроме того, установка бита выключения подтягивающих резисторов PUD регистра SFIOR отключает функцию подтягивания на всех выводах всех портов.

Ниже приведено описание порта ввода-вывода для универсального цифрового ввода-вывода. Большинство выводов портов поддерживают альтернативные функции встроенных периферийных устройств микроконтроллера. Обратите внимание, что для некоторых портов разрешение альтернативных функций некоторых выводов делает невозможным использование других выводов для универсального цифрового ввода-вывода.

Режим и состояние для каждого вывода определяется значением соответствующих разрядов трех регистров: DDxn, PORTxn и PINxn. Доступ к битам DDxn возможен по адресу DDRx в пространстве ввода-вывода и соответственно к битам PORTxn по адресу PORTx, а к битам PINxn по адресу PINx.

Биты DDxn регистра DDRx определяют направленность линии ввода-вывода. Если DDxn=1, то Pxn конфигурируется на вывод. Если DDxn=0, то Pxn конфигурируется на ввод. Независимо от значения бита направления данных DDxn состояние вывода порта может быть опрошено через регистровый бит PINxn. В табл. 2.4 подытоживается действие управляющих сигналов на состояние вывода.

Таблица 2.4 — Настройка вывода порта

DDxn	PORTxn	PUD (в SFIOR)	Ввод-вывод	Подтягивающий резистор	Комментарий
0	0	X	Ввод	Нет	Третье состояние (Z-состояние)
0	1	0	Ввод	Да	Pxn будет источником тока при подаче внешнего низкого уровня
0	1	1	Ввод	Нет	Третье состояние (Z-состояние)
1	0	X	Вывод	Нет	Вывод лог. 0 (втекающий ток)
1	1	X	Вывод	Нет	Вывод лог. 1 (вытекающий ток)

16-разрядный таймер-счетчик T1

16-разрядный таймер-счетчик T1 предназначен для точного задания временных интервалов, генерации прямоугольных импульсов и измерения временных характеристик импульсных сигналов.

Регистры таймера T1

Регистр таймера-счетчика (TCNT1), регистры порогов сравнения (OCR1A и OCR1B), а также регистр захвата (ICR1) являются 16-разрядными регистрами. В связи с этим, во время доступа к этим регистрам должна быть соблюдена специальная процедура. Чтобы записать данные в 16-разрядный регистр, необходимо сначала записать старший байт, а затем младший. А при чтении 16-разрядного регистра, наоборот, сначала считывается младший байт, а затем старший.

Регистры управления таймером TCCR1A и TCCR1B (табл. 2.5 и табл. 2.6) являются 8-разрядными регистрами, поэтому доступ к ним со стороны ЦПУ не связан с какими-либо ограничениями. Все сигналы запросов на прерывание представлены в регистре флагов прерываний таймеров (TIFR). Все прерывания индивидуально маскируются регистром маски прерываний таймеров (TIMSK).

Таймер-счетчик может тактироваться внутренне через предделитель или внешне тактовым источником, подключенным к выводу T1. Блок выбора тактового источника позволяет выбрать тактовый источник и фронт, по которому будет изменяться состояние таймера-счетчика. Если тактовый источник не задан, то таймер-счетчик находится в неактивном состоянии. Сигнал на выходе блока выбора тактового источника является тактовым сигналом таймера.

Значение регистров порогов сравнения (OCR1A и OCR1B) непрерывно сравнивается со значением счетчика. Результат сравнения может

использоваться для генерации прямоугольных импульсов с ШИМ или с переменной частотой на выходах OC1A и OC1B. В случае определения совпадения значений сравниваемых регистров устанавливается соответствующий флаг прерываний (OCF1A или OCF1B), который, в свою очередь, может служить источником прерывания.

Регистр захвата позволяет запомнить состояние таймера-счетчика при возникновении заданного внешнего события (фронт внешнего сигнала) на входе ICP или на выводах аналогового компаратора. На входе захвата фронта предусмотрена схема цифровой фильтрации (подавитель шума) для снижения риска срабатывания схемы захвата от помехи.

Таблица 2.5 — Формат регистра управления TCCR1A

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

Таблица 2.6 — Формат регистра управления TCCR1B

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

Назначение битов регистров управления:

COM1A1, COM1A0 — режим работы выходного сравнения A;

COM1B1, COM1B0 — режим работы выходного сравнения B;

WGM13, WGM12, WGM11, WGM10 — режим работы таймера/счетчика T1;

FOC1A, FOC1B — при записи в эти биты 1 моментально происходит событие выходного сравнения. Эти биты не работают в ШИМ-режимах;

ICNC1 — установка режима подавления шума на входе захвата. При сброшенном в состояние 0 бите ICNC1 функция подавления шума входного триггера захвата запрещена. Вход захвата переключается по первому нарастающему/падающему фронту, поступившему на вывод входа захвата. При установленном в состояние 1 бите ICNC1 выполняются четыре последовательных опроса состояния вывода и все четыре выборки должны иметь одинаковый (высокий/низкий), определяемый битом ICES1, уровень;

ICES1 — выбор фронта срабатывания на входе захвата. При сброшенном в состояние 0 бите ICES1 содержимое таймера/счетчика по падающему фронту на выводе входа захвата пересылается в регистр захвата входа ICR1. При установленном в 1 бите ICES1 содержимое таймера/счетчика пересылается в регистр захвата входа ICR1 по нарастающему фронту на выводе входа захвата;

CS12, CS11, CS10 — выбор источника тактовой частоты. Установкой состояния данных битов производится выбор источника тактового сигнала (в том числе коэффициента предварительного деления).

Режимы работы таймера T1

Под режимом работы 16-разрядного таймера понимается его алгоритм счета и поведение связанного с ним выхода формирователя импульсов, что определяется комбинацией бит, задающих режим работы таймера (табл. 2.7) и режим формирования выходного сигнала (табл. 2.9). В режимах с ШИМ биты COM1A (и точно так же COM1B) позволяют включить/отключить инверсию на генерируемом ШИМ-выходе (т.е. выбрать ШИМ с инверсией или ШИМ без инверсии). Для режимов без ШИМ эти биты определяют, какое действие необходимо выполнить при возникновении совпадения: сбросить, установить или инвертировать выход.

Таблица 2.7 — Выбор режима работы таймера/счетчика T1

Режим	WGM13	WGM12	WGM11	WGM10	Режимы модуляции	Модуль счета
0	0	0	0	0	Нормальный	0xFFFF
1	0	0	0	1	ШИМ ФК 8-bit	0x00FF
2	0	0	1	0	ШИМ ФК 9-bit	0x01FF
3	0	0	1	1	ШИМ ФК 10-bit	0x03FF
4	0	1	0	0	СТС	OCR1A
5	0	1	0	1	Быстрая ШИМ 8-bit	0x00FF
6	0	1	1	0	Быстрая ШИМ 9-bit	0x01FF
7	0	1	1	1	Быстрая ШИМ 10-bit	0x03FF
8	1	0	0	0	ШИМ ФЧК	ICR1A
9	1	0	0	1	ШИМ ФЧК	OCR1A
10	1	0	1	0	ШИМ ФК	ICR1A
11	1	0	1	1	ШИМ ФК	OCR1A
12	1	1	0	0	СТС	ICR1A
13	1	1	0	1	Зарезервировано	–
14	1	1	1	0	Быстрая ШИМ	ICR1A
15	1	1	1	1	Быстрая ШИМ	OCR1A

Таймер-счетчик T1 может использовать как внешний, так и внутренний тактовые сигналы (табл. 2.8).

Таблица 2.8 — Выбор источника тактового сигнала таймера/счетчика T1

CS12	CS11	CS10	Источник тактового сигнала
0	0	0	Stop условие — таймер/счетчик остановлен
0	0	1	CK
0	1	0	CK / 8
0	1	1	CK / 64
1	0	0	CK / 256
1	0	1	CK / 1024
1	1	0	Внешний тактирующий сигнал на выводе T1, спадающий фронт
1	1	1	Внешний тактирующий сигнал на выводе T1, нарастающий фронт

Таблица 2.9 — Режимы работы выходного сравнения A

COM1A1	COM1A0	Описание
Нормальный режим работы		
0	0	Таймер/счетчик отключен от вывода OC1A
0	1	Переключение выходной линии OC1A
1	0	Вывод сбрасывается в 0
1	1	Вывод устанавливается в 1
Режим быстрой ШИМ		
0	0	Таймер/счетчик отключен от вывода OC1A
0	1	В режиме 15 переключение выходной линии OC1A, иначе таймер/счетчик отключен от вывода OC1A
1	0	Очистка выходной линии OC1A при совпадении, установка при достижении верхнего предела
1	1	Установка выходной линии OC1A при совпадении, сброс при достижении верхнего предела
ШИМ ФК и ШИМ ФЧК		
0	0	Таймер/счетчик отключен от вывода OC1A
0	1	В режимах 9 или 11 переключение выходной линии OC1A, иначе таймер/счетчик отключен от вывода OC1A
1	0	Очистка выходной линии OC1A при совпадении во время счёта вверх, установка при совпадении во время счёта вниз
1	1	Установка выходной линии OC1A при совпадении во время счёта вверх, очистка при совпадении во время счёта вниз

Нормальный режим работы (Normal)

Самым простым режимом работы является нормальный режим. В данном режиме счетчик работает как обычный суммирующий счетчик. Переполнение счетчика происходит при переходе через максимальное 16-разрядное значение (0xFFFF) к нижнему пределу счета (0x0000). В нормальном режиме работы флаг переполнения таймера-счетчика TOV1 будет установлен на том же такте синхронизации, когда TCNT1 примет нулевое значение. В нормальном режиме можно использовать блок захвата. Блок сравнения может использоваться для генерации прерываний.

Режим сброса таймера при совпадении (CTC)

В режиме сброса таймера при совпадении разрешающая способность таймера задается регистрами OCR1A или ICR1. В режиме CTC происходит сброс счетчика (TCNT1), если его значение совпадает со значением регистра OCR1A (режим 4) или с ICR1 (режим 12). В данном режиме обеспечивается более широкий диапазон регулировки частоты генерируемых прямоугольных импульсов. Он также упрощает работу счетчика внешних событий. Временная диаграмма работы таймера в режиме CTC показана на рисунке 2.2. Счетчик (TCNT1) инкрементирует свое состояние до тех пор, пока не возникнет совпадение со значением OCR1A или ICR1, а затем счетчик сбрасывается.

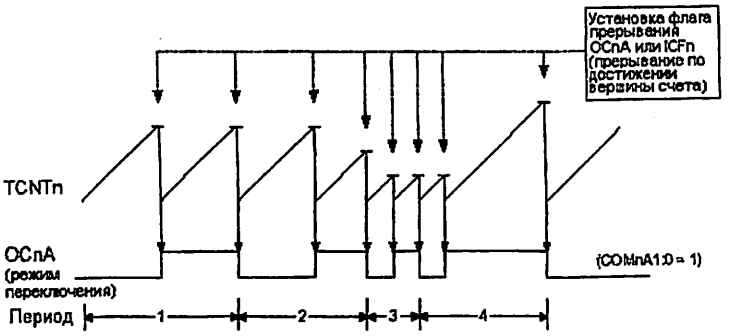


Рис. 2.2 — Временная диаграмма для режима CTC

Для генерации сигнала в режиме CTC выход OC1A может использоваться для изменения логического уровня при каждом совпадении, для чего необходимо задать режим переключения (COM1A1, COM1A0 = 0b01). Значение OC1A будет присутствовать на выводе порта, только если он настроен на выход. Частоту генерируемого сигнала можно определить по

формуле $f_{OC1A} = \frac{f_{clk}}{2 \cdot N \cdot (1 + OCR1A)}$, где переменная N задает коэффициент деления предделителя (1, 8, 32, 64, 128, 256 или 1024).

Режим быстрой ШИМ (Fast PWM)

Режим быстрой широтно-импульсной модуляции (ШИМ) предназначен для генерации ШИМ-импульсов повышенной частоты. В отличие от других режимов работы в этом используется однонаправленная работа счетчика. Счет выполняется в направлении от нижнего к верхнему пределу счета.

Если задан неинвертирующий режим выхода, то при совпадении TCNT1 и OCR1A сигнал OC1A устанавливается, а на верхнем пределе счета сбрасывается. Если задан инвертирующий режим, то выход OC1A сбрасывается при совпадении и устанавливается на верхнем пределе счета. За счет однонаправленности счета рабочая частота для данного режима в два раза выше по сравнению с режимом ШИМ с фазовой коррекцией, где используется двунаправленный счет. Возможность генерации высокочастотных ШИМ-сигналов делает использование данного режима полезным в задачах стабилизации питания, выпрямления и цифроаналогового преобразования. Высокая частота при этом позволяет использовать внешние элементы физически малых размеров (индуктивности, конденсаторы), тем самым снижая общую стоимость системы.

Разрешающая способность ШИМ может быть фиксированной: 8, 9 или 10 разрядов или задаваться регистром ICR1 или OCR1A, но не менее 2 разрядов (ICR1 или OCR1A = 0x0003) и не более 16 разрядов (ICR1 или OCR1A = 0xFFFF).

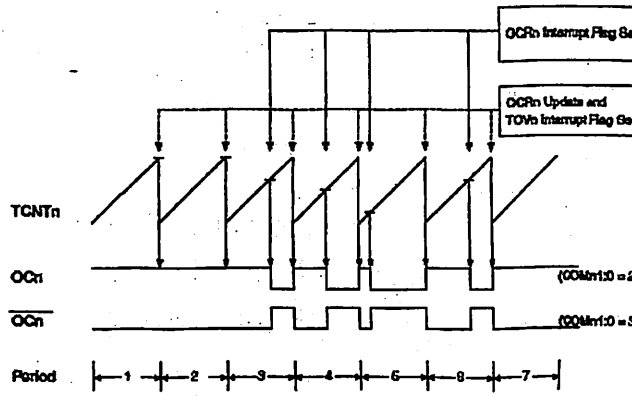


Рис. 2.3 — Временная диаграмма для режима быстрой ШИМ

Временная диаграмма для режима быстрой ШИМ представлена на рис. 2.3. На рисунке показан режим, когда для задания верхнего предела используется регистр OCR1A или ICR1. Значение TCNT1 на временной диаграмме показано в виде графика функции для иллюстрации однонаправленности счета. На диаграмме показаны как инвертированный, так и неинвертированный ШИМ-выходы. Короткой горизонтальной линией показаны точки на графике TCNT1, где совпадают значения OCR1A и TCNT1. Флаг прерывания устанавливается при совпадении. Флаг переполнения таймера-счетчика (TOV1) устанавливается всякий раз, когда счетчик достигает верхнего предела.

Рекомендуется использовать регистр ICR1 для задания верхнего предела, если верхний предел счета является константой. В этом случае также освобождается регистр OCR1A для генерации ШИМ-сигнала на выходе OC1A. Однако если частота ШИМ динамически изменяется (за счет изменения верхнего предела), то в этом случае выгоднее использовать регистр OCR1A для задания верхнего предела, т.к. он поддерживает двойную буферизацию.

Режим ШИМ с фазовой коррекцией (Phase Correct PWM)

Режим широтно-импульсной модуляции с фазовой коррекцией (ШИМ ФК) предназначен для генерации ШИМ-сигнала с фазовой коррекцией и высокой разрешающей способностью. Режим ШИМ ФК основан на двунаправленной работе таймера-счетчика. Счетчик циклически выполняет счет в направлении от нижнего предела (0x0000) до верхнего предела, а затем обратно от верхнего предела к нижнему пределу. При двунаправленной работе максимальная частота ШИМ-сигнала меньше, чем при однонаправленной работе, однако за счет такой особенности, как симметричность в режимах ШИМ с двунаправленной работой, данные режимы предпочитают использовать при решении задач управления приводами.

Разрешающая способность ШИМ в данном режиме может быть либо фиксированной (8, 9 или 10 разрядов), либо задаваться с помощью регистра ICR1 или OCR1A. Минимальная разрешающая способность равна 2-м разрядам (ICR1 или OCR1A = 0x0003), а максимальная — 16-ти разрядам (ICR1 или OCR1A = 0xFFFF). Временная диаграмма для режима ШИМ ФК представлена на рис. 2.4. На рисунке показан режим ШИМ ФК с использованием регистра OCR1A или ICR1 для задания верхнего предела. Состояние TCNT1 представлено в виде графика функции для иллюстрации двунаправленности счета. На рисунке представлены как неинвертированный, так и инвертированный ШИМ-выходы. Короткие горизонтальные линии указывают точки на графике изменения TCNTn, где возникает совпадение со значением OCRnx. Флаг прерывания устанавливается при возникновении совпадения.

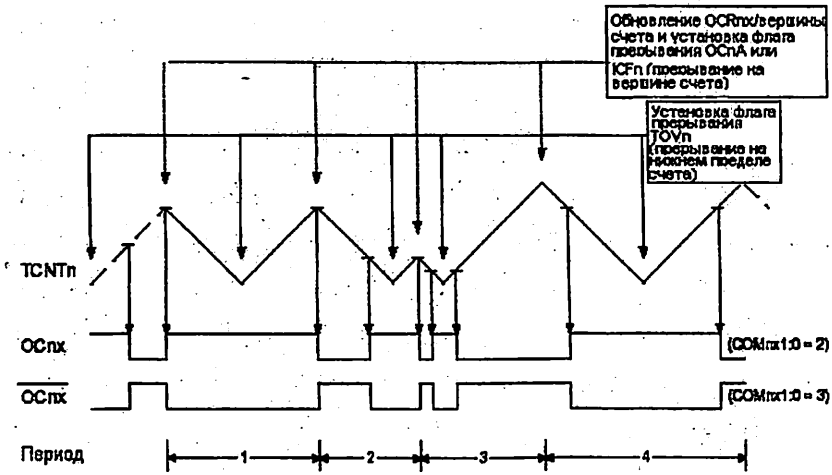


Рис. 2.4 — Временная диаграмма для режима ШИМ ФК

Флаг переполнения таймера-счетчика (TOVn) устанавливается всякий раз, когда счетчик достигает нижнего предела. Если для задания верхнего предела используется регистр OCRnA или ICRn, то соответственно устанавливается флаг OCnA или ICFn тем же тактовым импульсом, на котором произошло обновление регистра OCRnx из буферного регистра (на вершине счета). Флаги прерывания могут использоваться для генерации прерывания по достижении счетчиком нижнего или верхнего предела.

Если стоит задача изменения верхнего предела при работающем счетчике, то вместо этого режима рекомендуется использовать режим ШИМ ФЧК (фазовая и частотная коррекция). Если используется статическое значение верхнего предела, то между данными режимами практически нет отличий.

Режим ШИМ с фазовой и частотной коррекцией

Основное отличие между режимами ШИМ ФК и ШИМ ФЧК состоит в моменте обновления регистра сравнения OCR1A из буферного регистра. Разрешающая способность ШИМ в этом режиме может задаваться с помощью регистра ICR1 или OCR1A. Временная диаграмма для режима ШИМ ФЧК показана на рис. 2.5.

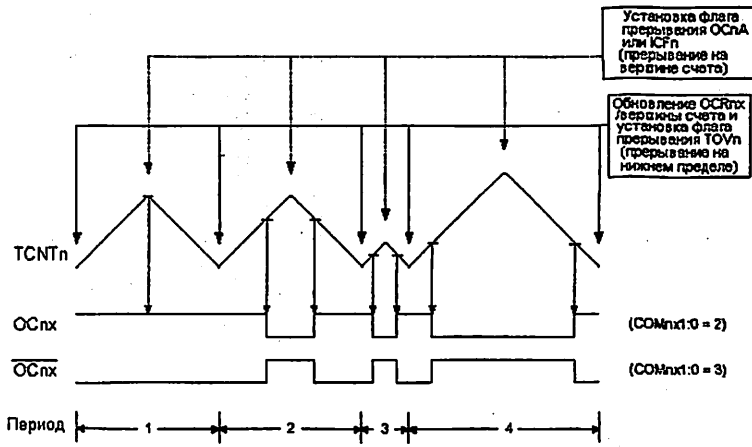


Рис. 2.5 — Временная диаграмма режима ШИМ с фазовой и частотной коррекцией

На рисунке 2.5 показано, что, в отличие от режима ШИМ ФК, генерируемый выходной сигнал симметричен на всех периодах. Поскольку регистры OCR1A обновляются на нижнем пределе счета, то длительности прямого и обратного счетов всегда равны. В результате выходные импульсы имеют симметричную форму, а следовательно, и откорректированную частоту.

Прерывания от таймеров /счетчиков

Для разрешения/запрещения прерываний от таймеров/счет-чиков T0, T1, T2 предназначен регистр TIMSK (табл. 2.10). Для разрешения какого-либо прерывания необходимо установить в 1 соответствующий разряд регистра TIMSK и, разумеется, флаг Iрегистра SREG. Для индикации наступления прерываний от таймеров/счетчиков T0, T1, T2 предназначен регистр TIFR (табл. 2.11).

Таблица 2.10 — Регистр масок прерываний TIMSK

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

OCIEnx — биты разрешения прерывания выходного сравнения;

TICIE1 — бит разрешения прерывания входного захвата;

TOIEn — биты разрешения прерывания по переполнению счётчика.

Таблица 2.11 — Регистр флагов прерываний от таймеров/счётчиков TIFR

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

OCFnx — флаги прерывания выходного сравнения;

ICF1 — флаг прерывания входного захвата;

TOVn — флаг прерывания по переполнению счётчика.

Программа работы

1. Рассчитав номер варианта (от N=1 до N=9), загрузить для отладки программу преобразования двоично-десятичного кода числа (байт) в двоичный (программа **decbin_to_bin**).

Перед выполнением программы необходимо в окне «Workspace» загрузить в регистр r16 число 10N+N в двоично-десятичном виде, которое подвергнется преобразованию.

```
;****** Программаdecbin_to_bin

    mov   r17,r16     ; Исходное двоично-десятичное число
                      ; хранится в r16
    andi  r17,0xF0    ; Выделение старшейтетрады (десятков)
    swap  r17         ;
    ldi   r18,10      ; Умножение десятков на десять
    mul   r17,r18     ;
    mov   r17,r16     ; Выделение младшей тетрады (единиц)
    andi  r17,0x0F    ;
    add   r17,r0      ; Сложение результатов. Результат остаётся в r17
```

При выполнении программы число копируется в регистр r17, там на него накладывается маска 0b11110000 при помощи команды логического умножения (andi), которая выделяет десятки десятичного кода. После этого результат перемещается в младшую тетраду и умножается на 10. Результат умножения сохраняется в регистровой паре r1:r0. Так как наше число не может быть больше 99, то оно целиком поместится в младший байт результата, т.е. в r0. После этого повторно загружаем исходное число в r17, ещё раз накладываем маску, только на этот раз для выделения младшей тетрады, и суммируем результат с содержимым регистра r0, в котором хранились десятки. Конечный результат будет представлен в регистре r17.

Проследить процесс выполнения программы в пошаговом и автоматическом режимах. Пояснить содержимое строк окна «Disassembler».

Внести ошибки в программу (неверная мнемоника команды, неверный операнд и т.п.) и проследить сообщения ассемблера при этом в окне **ViewOutput.**

Модифицировать программу в соответствии со своим вариантом индивидуального задания:

1. Преобразовать дополнительный код числа (байт) в прямой.
2. Преобразовать двоичный код (от 0 до 99) в двоично-десятичный.
3. Просуммировать два числа в двоичном коде. Сумму, большую 255, заменить байтом единиц.
4. Сложить два двухбайтовых числа.
5. Вычесть два числа в двоичном коде. Разность, меньшую нуля, заменить байтом нулей.
6. Умножить на два двухбайтовое содержимое регистров R9..R10 (меньшее 32 000).
7. Сложить два десятичных числа (байт) в двоично-деся-тичном коде.
8. Реализовать суммирующий двоично-десятичный счетчик.
9. Реализовать вычитающий двоично-десятичный счетчик.

2. Загрузить для отладки программу определения максимального элемента массива 8-разрядных чисел SRAM (программа**max_el_mass).**

Перед выполнением программы необходимо заполнить область памяти данных случайными числами, начиная с адреса 0x60. Количество элементов определяется числом, загружаемым в начале программы в r18.

Для обращения к памяти с адресом более 8 разрядов программа использует специально для этого предназначенныйZ-ре-гистр. Он состоит из пары регистров r31:r30. В начале выполнения программа копирует первый элемент массива в r16, а адрес этого элемента в X (регистровая пара r27:r26). После этого каждый элемент сравнивается со значением в r16, и в том случае, если элемент окажется больше содержимого r16, он замещает собой предыдущее значение в r16, а также адрес нового наибольшего элемента

копируется в пару X. Каждый раз при сравнении из количества элементов вычитается единица. Как только содержимое r18 станет равным 0, выполнение программы можно прекращать в связи с перебором всех элементов. В итоге получим наибольший элемент в r16, его адрес в X.

```
;****** Программа max_el_mass

        ldi r30,low($100)  ;Загрузка в регистр Z начального адреса
        ldi r31,high($100) ;массива чисел
        ldi r18,12         ;Загрузка количества элементов массива
        ld r16,z           ;Загрузка в регистр r16 первого элемента массива
        mov r26,r30        ;Загрузка в X адреса первого элемента
        mov r27,r31
        dec r18
m1:     inc r30            ;Увеличение Z на единицу для загрузки
                           ; следующего элемента
        ld r17,z           ;Загрузка элемента для сравнения
        cp r16,r17         ;Сравнение максимального (или первого) элемента
                    ;с только что загруженным
        brsh m2
        mov r16,r17        ;его перезапись в r16 в случае, если больше,
        mov r26,r30        ;и сохранение адреса в X
        mov r27,r31
m2:     dec r18            ;Уменьшение счётчика количества элементов
        brbc 1,m1          ;Если не все элементы перебраны, переход на m1
m3:     jmp m3             ;Зацикливание по завершении
```

Модифицировать программу в соответствии со своим вариантом индивидуального задания:

1. Определить минимальный элемент массива SRAM.

2. Сформировать массив 3, элементы которого определяются как разность соответствующих элементов массивов 1 и 2. Отрицательную разность заменить нулем.

3. Сформировать массив 3, элементы которого определяются как сумма соответствующих элементов массивов 1 и 2. Сумму, большую 255, заменить байтом единиц.

4. Количество одинаковых элементов массивов 1 и 2 поместить в регистр R0.

5. Количество чисел массива 1, совпадающих с содержимым регистра R1, поместить в регистр R0.

6. Двоичные числа массива преобразовать в двоично-десятичные.

7. Двоично-десятичные числа массива (меньшие 99) преобразовать в двоичные.

8. Сформировать массив 2, элементы которого представляют дополнительный код восьмиразрядных чисел со знаком массива 1.

9. Отсортировать массив по возрастанию элементов.

3. Набрать и отладить прикладную программу **fairy**, позволяющую получить эффект бегущей 1 на линиях порта.

Ввести код программы в отладчик AVRStudio и проверить ее работу в пошаговом режиме. Проследить изменения, происходящие в регистрах SREG, PORTB и PINB по мере выполнения программы. Для чего в регистр DDRB заносится 0xFF? Чем отличается команда rol от команд lsl и asr?

;****** **Программа fairy**

```
.INCLUDE "C:\Program Files\Atmel\AVR
    Tools\AvrAssembler\Appnotes\m8def.inc"
    ; Подключение файла определения адресов

            ldi   r16,0xFF
            out   DDRB,r16
            sec
            clr   r16
m1:         rol   r16
            out   PORTB,r16
            rjmp  m1
```

Дополните программу подпрограммой задержки «Delay» таким образом, чтобы содержимое r17 определяло длительность паузы между сменой состояния на выводах порта.

Примечание: для вызова подпрограммы необходимо активировать стек, что происходит автоматически при указании его начала в паре регистров sph:spl.

Модифицировать программу в соответствии со своим вариантом индивидуального задания и произвести ее отладку:

1. Мультивибратор (тетрады порта D заполняются либо единицами, либо нулями в цикле, скорость миганий можно изменять с помощью порта B).

2. Бегущий огонек со сменой направления на линиях порта C.

3. Елочка 1 (линейка светодиодов, подключенная к линиям порта B, последовательно заполняется огнями и затем гаснет, после чего эффект периодически повторяется).

4. Бегущий огонек на линиях порта C должен сменить направление, если на пяти линиях порта B установлены логические единицы.

5. Елочка 2 (линейка светодиодов, подключенная к линиям порта, последовательно заполняется огнями и постоянно горит, а звезда — старший бит — моргает).

6. Если на выводы порта D пришло число с нечётным количеством единиц, то оно передаётся через порт B, если с чётным — через порт C.

7. «Бегущий огонёк» в одну сторону бежит по выводам порта B, в другую — порта D.

8. Одна тетрада поступающего на выводы порта D числа должна быть отправлена через порт B, другая через C.

9. Если поступившее на выводы порта D число чётное, то должны «мигать» выводы порта B, если нет — то порта C.

4. Загрузить для отладки программу генерации сигнала заданной частоты(программа**Generator**).

```
;****** Программа Generator
; Генератор импульсов с программируемым периодом T=2n(1+X), где
; X — число в регистре OCR1A, n — коэффициент деления предделителя

.INCLUDE  "C:\Program Files\Atmel\AVR
          Tools\AvrAssembler\Appnotes\m8def.inc"
      ldi       r16,0x02          ; Линию PB1 на вывод
      out       DDRB,r16          ;
      ldi       r16,0b01000000    ; Режим CTC таймера T1, состояние
                                  ; вывода PB1 при сравнении меняется
      out       TCCR1A,r16        ; на противоположное
      ldi       r16,0b00001001    ; Режим CTC таймера T1 на частоте
      out       TCCR1B,r16        ; тактирования  (n=1)
      ldi       r16,99            ; Модуль счета X=99 для
      out       OCR1AL,r16        ; периода T=200 тактов
m1:   rjmp      m1
```

Изменить программу так, чтобы период генерируемых прямоугольных импульсов на выводе PB1 составил N секунд при использовании кварцевого резонатора на 16 МГц (N — вариант задания).

5. Составить комментарий к программе **PWM2.**

```
;****** Программа PWM2
.INCLUDE  "C:\Program Files\Atmel\AVR
           Tools\AvrAssembler\Appnotes\m8def.inc"
          ldi    r16,0x02
          out    DDRB,r16
          ldi    r16,0x7F
          out    OCR1AL,r16
          ldi    r16,0b11000001
          out    TCCR1A,r16
          ldi    r16,0b00000010
          out    TCCR1B,r16
    m1:   rjmp   m1
```

Ввести код программы в отладчик AVRStudio и проверить ее работу в пошаговом режиме. Какой режим ШИМ выбран в данном случае? Какова относительная длительность импульсов на выводе OC1A? С какой частотой относительно частоты кварца поступают импульсы синхронизации на таймер/счётчик?

Контрольные вопросы

Чем ограничен размер массива?

Перечислите все методы адресации памяти данных.

Разрешены ли прерывания после системного сброса?

Для чего нужен регистр Inputcapture (ICR)? Назовите ситуации, когда необходимо его использование. Назовите его альтернативные функции.

Что должно произойти при достижении счётчиком значения 0x01FF, если в битах WGMn3:1 записано 0b0010? 0b0110? Как при этом поведёт себя OCnx?

Что может выступать в качестве источника импульсов синхронизации таймеров/счётчиков?

Содержание отчета

Отчет должен содержать листинги отлаживаемых программ (в том числе и по индивидуальным заданиям), комментарии по ходу выполнения пунктов работы и рисунки, вставляемые в текст формата WORD, отображающие окна регистров и памяти, а также ответы на контрольные вопросы.

Лабораторная работа №3

МОДЕЛИРОВАНИЕ РАБОТЫ МИКРОКОНТРОЛЛЕРА AVR С ПОМОЩЬЮ СИМУЛЯТОРА VMLAB

Цель работы. Целью лабораторной работы является отладка прикладных программ на языке Си для микроконтроллера AVR с помощью компилятора CVAVR и симулятора VMLAB.

Программа работы

1. Установите в директорию **C:\CVAVR** свободную версию компилятора **CodeVisionAVR**. В директории C:\CVAVR создайте папку **z1** (задача 1) для файлов первого проекта.

Запустите компилятор. Для создания файла проекта нажимайте: **Файл -> новый -> проект ->ОК ->No**

- перейдите в созданную для проекта папку **z1** и введите в поле «имя файла»: **z1**;
- нажмите «сохранить» — откроется окно конфигурации проекта;
- перейдите на закладку «C compiler»;
- выберите МК (Chip) ATmega16;
- установите частоту тактирования МК (Clock) 4.0 МГц;
- нажмите **ОК**.

Перед вами появится открытый текстовый файл **ProjectNotes — z1.prj**, в котором вы можете записывать свои замечания и мысли по проекту.

Теперь нужно создать главный для нас текстовый файл для набора исходного текста на Си — его расширение.c

нажимайте:

Файл ->New ->Source ->ОК

появился файл **untitled.c**

нажимайте:

Файл — Сохранить как

введите в поле «имя файла»: **z1.c и нажмитеСохранить.**

Нужно добавить созданный файл z1.c в список файлов проекта — откройте меню конфигурирования проекта: **Project ->Configure.**

В открывшемся диалоге, нужно выбрать ярлык «Files» и нажать кнопку «**Add**». В новом диалоге выберите файл «**z1.c**» и нажмите «**Открыть**». Теперь файл включен в проект:

нажимайте **ОК**

максимизируйте (разверните) окно файла — **z1.c**

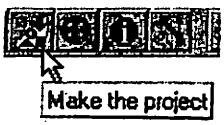
Теперь все готово к собственно программированию, т.е. к созданию текста программы на языке Си. Ниже в таблице подготовлен текст программы к задаче 1, реализующей следующее техническое задание: разработать устройство на микроконтроллере ATmega16, которое будет отображать в двоичном виде горящими светодиодами 8-битное число начиная с 0 и с постоянным увеличением на 1. Устройство питается постоянным стабилизированным напряжением от 4 до 5.5 вольт. Тактирование МК осуществляется от кварцевого резонатора с частотой 4 МГц. Всего подключено 8 светодиодов от ножек порта A через токоограничительные резисторы к питанию МК. Переключение светодиодов должно производиться с паузами в 65 мс.

```
#include<mega16.h> /* Вставить вместо этой строки текст файла
mega16.h, содержащий описание регистров МК */
#define PA_OUT DDRA = 0xFF
/* Заменить везде в тексте программы
PA_OUT   на     DDRA = 0xFF */
 // ++++ функция инициализации МК ++++
voidinitialization(void){
```

```
PA_OUT;//сделать весь PORTA выходом
TCCR0 = 0x05;/* таймер включить считать, делая один отсчет
каждые 1024 колебания на ножке XTAL1 */
}
Charper=0;
// ++++ Главная функция ++++
voidmain (void){
initialization(); /* Вызвать функцию инициализации МК — т.е.
настройки нужных нам устройств МК в соответствии с поставленной
задачей */
//Бесконечный цикл
while (1){//Делать всегда
PORTA=~(per++);
while (!(TIFR&0x01));
// ждем установки флага переполнения timer0
TIFR = 0x01;
// очистить флаг переполнения timer0
    }; //цикл закончен
} //скобка для main()
```

Запишите (без комментариев) программу в окно исходного текста программы. Сохраните изменения: **файл ->SaveAll.**

Для компиляции программы нажмите кнопочку **«Maketheproject»**



Загляните в папку нашего проекта — z1. В результате компиляции там появилось много новых файлов. Главные для нас:

z1.hex — файл-прошивка для «загрузки» в МК;

z1__.c — копия файла z1.c для симуляторов;

z1.cof — информация, связывающая содержимое файлов z1__.c и z1.hex. Эта информация позволяет при симуляции в VMLAB наблюдать движение программы прямо по коду на языке Си. Указанные файлы будем использовать в симуляторе VMLAB. Необходимым для реального МК является лишь файл прошивки.

Следующие четыре файла содержат нашу программу, написанную на стандартном ассемблере для AVR с привязкой к тексту на Си: z1.asm, z1.lst, z1.vec, z1.inc. Остальные файлы практически не интересны.

2. Запустите VMLAB и откройте созданный проект:

Project ->OpenProjectFile

Перейдите в папку задачи 1 C:\CVAVR\z1\ и наберите имя файла z1_vm.prj проекта для VMLAB. После появления фразы, что такой файл не существует, VMLAB предложит создать его, с чем вы соглашайтесь. В появившемся окне запишите без комментариев приведенный ниже в таблице текстовый файл.

```
; Файл-проект z1_vm.prjдля симуляции по задаче 1.
; Комментарии пишутся в VMLAB только в одну строчку
; после точки с запятой
; МК как бы «прошит» файлом — z1.hex. После включения МК
; горящие светодиоды показывают в двоичном виде числа от 0
; до 255 и далее опять с нуля и так по кругу...
; светодиоды подключены к порту_A МК

.MICRO «ATmega16» ; симулируемый МК
.TOOLCHAIN «GENERIC»
.TARGET «z1.hex» ; что «прошито» в МК
```

```
.COFF «z1.cof»
.SOURCE «z1__.c»
.POWER VDD=5 VSS=0 ; Питание +5 вольт
; VSS это GND МК — «общий» провод схемы
; Относительно него измеряются напряжения
.CLOCK 4meg ; частота кварца 4 МГц
; Точнее, это частота тактирования МК
; Ввод схемы устройства по задаче 1
; 8 светодиодов подключаются катодами через резисторы
: номиналом 560 Ом к ножкам МК с 33 до 40
; резистор R1 подключить к узлу D1_NODE и к выводу PA0 МК
; анод светодиода к цепи +5 В.
Остальные 7 светодиодовподключаются аналогично

D1 VDD D1_NODE
R1 D1_NODE PA0 560

D2 VDD D2_NODE
R2 D2_NODE PA1 560

D3 VDD D3_NODE
R3 D3_NODE PA2 560

D4 VDD D4_NODE
R4 D4_NODE PA3 560

D5 VDD D5_NODE
R5 D5_NODE PA4 560
```

```
D6 VDD D6_NODE
R6 D6_NODE PA5 560

D7 VDD D7_NODE
R7 D7_NODE PA6 560

D8 VDD D8_NODE
R8 D8_NODE PA7 560

; Сигналы на ножках PA0 PA1 PA2
; будем наблюдать в окне виртуального осциллографа — «SCOPE»

.PLOT V(PA0) V(PA1) V(PA2)
; Рисовать графики напряжения в перечисленных узлах схемы
```

ВменюProject запуститеRe-Build all ...

Через меню View откройте два компонента: SCOPE — это виртуальный запоминающий осциллограф симулятора и ControlPanel — это панель, на которой содержатся нужные нам светодиоды и многое другое, пока нам не нужное.

Через меню Window откройте (обычно оно открывается сразу при открытии проекта) окно Code — в этом окне вы увидите текст симулируемой программы.

Обратите внимание на окно Messages — в нем появляются служебные сообщения симулятора по ходу работы. В окне Messagesдолжно появиться сообщение об успехе и что все готово к запуску (Success!Allreadytorun). Кроме того, на панели инструментов загорится зеленый светофор — это кнопка, которой можно запускать симуляцию.

Нажатие зеленого светофора эквивалентно подаче «1» на вывод RESET МК при включенном питании, но еще не выполнявшем программу.

В окне Scopeпоявились три графика для сигналов, которые мы будем наблюдать. Установите масштаб по вертикали 2 вольта на деление, а по горизонтали 50 мс.

В окне Codeпоявилось серое поле слева и зеленые квадратики напротив исполняемых строк кода программы на Си — кликнув по такому квадратику, мы можем поставить точку останова программы.

Разместите три окна и ControlPanelна экране компьютера так, чтобы видеть их все.

Нажмите «светофор» для запуска симуляции программы.

Программа запустится и остановится — в окне Messages появится сообщение. Опять нажимаем на «светофор». Симулятор опять останавливается и сообщает, что произошел сброс от «сторожевого таймера МК» — мы не указали симулятору, что не используем его. Опять нажимаем на «светофор» — теперь программа будет работать непрерывно, пока мы ее не остановим.

Пусть программа симулирует, а вы понаблюдайте за тем, что происходит в указанных выше окнах. Что отображается в окне ControlPanel, кроме светодиодов?

Понаблюдайте за окнами **SCOPE** и **Code** и за светодиодами. В окне **Code** при симуляции возникают и растут желтые полосы, подсвечивающие строки исполняемой программы. Длины этих подсветок пропорциональны времени, в течение которого программа выполняет код этих строк.

Какой ток потребляется микроконтроллером от источника питания? Остановите симуляцию, нажав красный восьмиугольник «Стоп» и измерьте длительность периода импульсов на ножке PA2 МК. Насколько соответствует она расчетной величине? Для измерения временного промежутка в окне SCOPE симулятора VMLAB нужно установить вертикальные курсоры 1 и 2 на границах измеряемого интервала, и в поле Cursordeltatime появится значение времени между двумя курсорами.

При измерении коротких повторяющихся интервалов можно мерить время сразу нескольких, а результат поделить затем на число таких интервалов между измерительными курсорами.

Перезапустите МК, кликнув по кнопке с **круговой темно-синей стрелкой**. Вы как бы отключаете и затем снова подаете питание на МК, но создаете «0» на ножке RESET МК — вследствие чего программа не стартует!

Какую функцию выполняет команда **PORTA=~(per++);** ?

Приведите в отчете схему подключения светодиодов к МК.

3. Модифицируйте программу. Переключите светодиоды к порту С. Время паузы между переключениями светодиодов уменьшить в 2 раза.

Для изменения Си кода программы просто запустите компилятор CodeVisionAVR (VMLAB выключать не нужно!) и внесите нужные изменения, затем откомпилируйте проект. Далее перейдите в VMLAB, сделайте **глубокий рестарт** и затем **Re-buidall.** Все! Изменения внесены, и все опять готово к симуляции. Таким образом, компилятор и симулятор работают одновременно в одной папке проекта и не мешают, а помогают друг другу. В отчет включите файлы z1.c и z1_vm.prj модифицированного проекта.

4. В следующем проекте будем выводить данные на символьный LCD-дисплей (жидко-кристаллический индикатор). Схема его подключения к порту А микроконтроллера приведена на рис. 3.1 (там же указан источник информации, в котором вы можете более подробно ознакомиться с решаемой задачей).

Запустите компилятор **CodeVisionAVR**, затем генератор начального кода **«CodeWizardAVR»** — кликнув **серую шестеренку** слева от красного жучка... Выберите ATmega16 и частоту кварца 4 МГц. Перейдите к закладке LCD и укажите PORTA и 16 символов.

Выполнив **Файл ->Generate, SaveandExit**, создайте в директории C:\CVAVR папку **z2** (задача 2) для файлов нового проекта. Сохраните,

нажимая три раза z2, файлы z2.c, z2.prj и z2.cwp. Посмотрите сгенерированный мастером файл начального кода программы **z2.c**. Какими командами проводится инициализация LCD-дисплея? Можно ли удалить из программы команды, реализующие инициализацию периферийных устройств, не используемых в данной задаче?

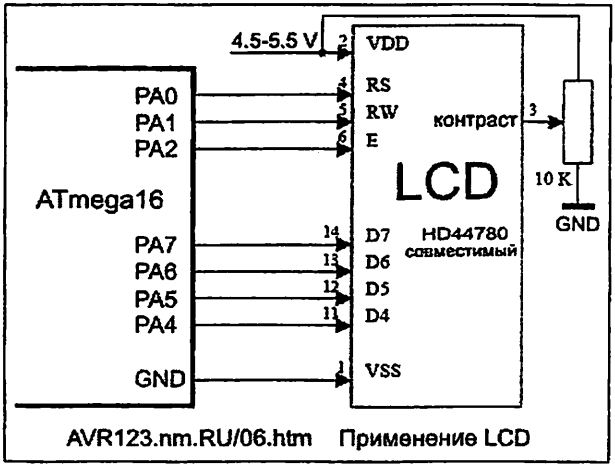


Рис. 3.1 — Типовая схема включения LCD-дисплея

После команды

lcd_init(16); // LCD 16 символов на строку

добавьте две строчки:

lcd_gotoxy(5,0); // вывод символов с 6-й позиции в первой строке

lcd_putsf("Hello!"); //счет строк и символов начинается с нуля!

Сохраните (File ->SaveAll) и откомпилируйте программу.

Не закрывая компилятор, откройте **VMLAB**. В окне **OpenProjectFile** впишите имя файла z2_vm и откройте файл проекта для симулятора **z2_vm.prj**. Впишите в него приведенный ниже в рамке текст и запустите **Rebuildall** ... Загоревшийся светофор говорит о том, что программа готова к симуляции. Откройте окно ControlPanel и, трижды нажав светофор, добейтесь непрерывной симуляции. Долгожданная надпись на экране

LCDпоявится не сразу (процесс инициализации LCD продолжается достаточно долго). Почему через некоторое время загорается светофор?

; файл **z2_vm.prj**

.MICRO "ATmega16"

.TOOLCHAIN "GENERIC"

.TARGET "z2.hex"

.COFF "z2.cof"

.CLOCK 4meg

Xdisp LCD(16 2 250K) PA0 PA1 PA2 PA7 PA6 PA5 PA4 nc3 nc2 nc1 nc0

Не закрывая **VMLAB**,вернитесь в компилятор **CVAVR**. После команды **#include<mega16.h>**добавьте команду

#include<delay.h> // функции организации задержек

После команды **lcd_putsf("Hello!");**добавьте команды:

delay_ms(200);

lcd_clear(); // очистка экрана LCD

delay_ms(200);

lcd_gotoxy(5,1);

lcd_putsf("FINISH!");

В последнем цикле программы перед комментарием // **Placeyourcodehere**добавьте команду **#asm("wdr")** и перекомпилируйте проект.

Вернитесь в VMLAB. Сделайте глубокий рестарт и запустите **Re-buildall** ... Как теперь выводится информация на табло дисплея? Почему не загорается светофор после запуска непрерывной симуляции?

5. Проведите исследование работы АЦП. В папке C:\CVAVR \z3 с помощью компилятора создайте файлы проекта задачи 3 на базе программы**z3.c**, текст которой приведен ниже в рамке

```
// файлz3.c
#include <mega16.h>
#include <delay.h>

interrupt [ADC_INT] void adc_isr(void) {
PORTB=(char)~(ADCW>>2);
delay_ms(20);
ADCSRA|=0x40; }

void main(void) {
PORTB=0xFF;
DDRB=0xFF;
ADCSRA=0x8E;
asm("sei")
ADMUX=0;
ADCSRA|=0x40;
while (1); }
```

Затем с помощью симулятора запишите файл Z3_vm.prj.

```
; файл Z3_vm.prj
.MICRO "ATmega16"
.TOOLCHAIN "GENERIC"
.TARGET "z3.hex" ; эмулируемая прошивка МК
.COFF "z3.cof" ; файл содержит привязку
; содержимого [.hex] к коду в [__.c]
.SOURCE "z3__.c" ; исходник на Си, на который сориентирован файл
[.cof].
; это CodeVision добавляет '__' при компиляции
```

.TRACE ; выводить отладочную информацию в окне
; SCOPE — розовым (см. HELP эмулятора)
.CLOCK 4meg ; частота используемого кварца

; Обозначения точек МК, к которым можно
; «подключить» эмулятор: RESET, AREF, PA0—PA7, PB0—PB7, PC0—PC7, PD0—PD7, ACO, TIM1OVF
; Для использования АЦП МК нужно подать опорное напряжение на вывод AREF — мы подадим 5 вольт питания МК. Но в VMLAB
; нельзя соединить два узла напрямую. Берем резистор на 1 Ом.

R1 VDDAREF 1 ; резистор R1 подключен к
; узлам VDD и AREF через сопротивление 1 Ом

; опорное напряжение Vref у нас 5 вольт —
; значит, при подаче 5 вольт на вход АЦП
; мы получим результат: 1111111111 (АЦП 10-разрядный)

; Вход0 АЦП (это вывод PA0 МК) мы подключим к
; подвижному контакту переменного резистора
; (Slider 1 в окне «ControlPanel») —
; чтобы при эмуляции менять напряжение на входе АЦП.

V1 PA0 VSS SLIDER_1(0 5)
; на концах переменного резистора 0 и 5 вольт

; Эмулятор имеет 8 светодиодов —
; подключаем их к выводам порта B

D1 VDD PB0

D2 VDD PB1

D3 VDD PB2

D4 VDD PB3

D5 VDD PB4

D6 VDD PB5

D7 VDD PB6

D8 VDD PB7

; Эмулятор допускает прямое подключение светодиодов к

; плюсу питания и выводам МК — в действительности необходим

; токоограничительный резистор 430—910 Ом

; последовательно с каждым светодиодом!

.PLOT V(PA0) ; на экран осциллографа (окно «SCOPE»)

; выведем напряжение на движке потенциометра

Запустив проект на симуляцию, понаблюдайте за светодиодами и осциллографом, изменяя положение движка потенциометра. Какое напряжение соответствует единице младшего разряда АЦП? Раскройте окно Peripherals и понаблюдайте за регистрами АЦП при изменении положения движка потенциометраS1. Сравните показания светодиодов и содержимое регистров ADCH и ADCL.

Просмотрите содержимое памяти программ и текст программы на ассемблере. Сколько ячеек занимает программа? По какому адресу расположен вектор прерывания по завершению процесса аналого-цифрового преобразования?

6. Запустите на симуляцию проект, подготовленный в папке **z4** (задача 4). Проект реализован на МК ATmega16.

В окне **SCOPE** (это виртуальный осциллограф) можно увидеть изменения напряжений на ножках МК, указанных в файле проекта — **vmlab.prj**. Верхняя осциллограмма — это сигнал на ножке **TXD (PD1)** последовательного порта **USART**, по которой МК передает данные на COM-порт ПК через интерфейс RS232, что передает МК, мы видим в виртуальном терминале **TTY** панели **ControlPanel**. Там выводится значение **ШИМ (PWM)**-сигнала, создаваемого на ножке PD5. Сам сигнал виден в окне **SCOPE** — посмотрите, как он меняется в соответствии с сообщаемыми числовыми значениями. На ножке**PD4** формируются импульсы той же частоты с неизменной длительностью.

В файле проекта **vmlab.prj** к ножке **PD5** подключен простейший фильтр нижних частот (ФНЧ) из резистора и конденсатора — он преобразует **ШИМ-сигнал** в постоянное напряжение, которое можно увидеть в окне **SCOPE** (сигнал **DAC**).

Формат передачи данных в примере — 8N1 (это формат по умолчанию для ПК). В таком формате передача байта начинается со «старт-бита» — это лог. «0» на ножке TXD для USART МК и +5...+15 В для COM-порта ПК. Затем на ножку TXD выводятся все 8 бит передаваемого байта, начиная с нулевого. За время передачи бита приемник должен определить и запомнить этот уровень. Далее идет «стоп-бит» — это лог. «1» на ножке TXD для USART МК и –5...–15 В для COM-порта ПК. Для согласования уровней между МК и ПК включают адаптер MAX232.

7. Протестируйте работу программы, текст которой приведен ниже. Разработайте программу, реализующую световой эффект бегущего огонька без использования ассемблерных вставок.

```
#include <mega16.h>
#include <delay.h>
void main(void){
DDRB=0xFF;
```

```
#asm ("ldi r20,1")
while(1){
delay_ms(10);
#asm ("lsl r20")
#asm ("out 0x18,r20")
if (PORTB==0){
PORTB++;
#asm ("ldi r20,1")};
};}
```

Контрольные вопросы

1. Назовите нагрузочную способность линий портов AVR.
2. Какими ассемблерными вставками можно разрешать и запрещать глобально прерывания в программе для AVR на языке Си?
3. Запишите результат выполнения арифметических операций: 245/37 и 245%37.
4. Какими командами можно организовать задержку в одну секунду в программе для AVR на языке Си?
5. Дать комментарий к команде PORTA=~(per++);
6. Объявите переменную mnogo, если она может принимать значения от нуля до миллиона.
7. Прокомментировать результат выполнения команды ADCSRA|=0x40;

Содержание отчета

Отчет должен содержать тексты отлаживаемых программ с конкретной датой их компиляции, комментарии по ходу выполнения пунктов программы работы и рисунки, вставляемые в текст формата WORD, отображающие окна **SCOPE , ControlPanel** и т.д. с результатами моделирования, а также ответы на контрольные вопросы.

СПИСОК ЛИТЕРАТУРЫ

1. Щелкунов Н.Н., Дианов А.П. Микропроцессорные средства и системы.– М.: Радио и связь, 1989.– 288 с.

2. Сташин В.В., Урусов А.В., Молгонцева О.Ф. *Проектирование* цифровых устройств на однокристальных *микроконтроллерах*.– М.: Энергоатомиздат, 1990.– 224 с.

3. Однокристальные микро-ЭВМ.– М.: МИКАП, 1994,– 400 с. (Справочник)

4. Гребнев В.В. Однокристальные микро-ЭВМ (микроконтроллеры) семейства MCS-96.– Псков.: Псковская коммерческая палата, 1995.

5. Однокристальные микроконтроллеры / Л.К.Головина, И.С.Зуев, М.С.Куприянов, М.Г.Пантелеев, Г.А.Петров, Д.В.Пузанков. Учебное пособие. СПб.: Изд-во СПбГЭТУ "ЛЭТИ", 1999, 76 с.

6. Микропроцессорные системы./ М.С. Куприянов, Р.И. Грушвицкий, О.Е. Мартынов и др. Под. ред Д.В. Пузанкова Учебное пособие для вузов. - СПб, Политехника, 2002, 936 с.

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

8. Справочники по микропроцессорным системам под редакцией Хвоща С.Т. или Шахнова.

9. Микроконтроллеры AVR — самоучитель начинающим с нуля: Краткий курс [электронный ресурс]. Режим доступа: http://avr123.nm.ru

10. www.atmel.com.

11. www.gaw.ru

12. www.msclub.ce.cctpu.edu.ru/MCU_MPU/AVR/Architec.htm

13. www.atmel.ru/Articles

ОГЛАВЛЕНИЕ

Формат 60x84 1/16
Заказ № - 109. Тираж - 30

Отпечатано в Издательско полиграфическом центре «ALOQACHI» при ТУИТ
Ташкент ул. Амир Темура, 108