МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (СГАУ)

Микроконтроллер AVR ATMega 128. Справочные сведения

Методические указания к лабораторным работам

Самара
2011

УДК 004.384 (004.431.4, 681.32)
М 597

Составитель: **Зайцев Александр Анатольевич**

Микроконтроллер AVR ATMEGA 128. Справочные сведения: метод. указания к лабораторным работам / М-во образования и науки РФ, самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т); сост. А. А. Зайцев. - Электрон. текстовые и граф. дан. (1,8 Мбайт). - Самара, 2011. - 1 эл. опт. диск (CD-ROM).

Режим доступа: http://ivt.ssau.ru/course/view.php?id=12

Содержат описание, характеристики основных узлов микроконтроллеров AVR семейства Mega. Даны рекомендации для программирования микроконтроллера и управления его периферийными устройствами. Приведены описания директив ассемблера и сводная система команд.

Методические указания предназначены для бакалавров направления 162500.62 «Техническая эксплуатация авиационных электросистем и пилотажно-навигационных комплексов» факультета инженеров воздушного транспорта, выполняющих практические и лабораторные работы по дисциплине «Бортовые цифровые вычислительные устройства», изучаемой в 7 семестре.

Разработано на кафедре эксплуатации авиационной техники.

Порты ввода/вывода.

Каждый порт микроконтроллеров состоит из определенного числа выводов, через которые микроконтроллер может осуществлять прием и передачу цифровых сигналов. Задание направления передачи данных через любой контакт ввода/вывода может быть произведено программно в любой момент времени. Входные буферы всех выводов построены по схеме триггера Шмитта. Для всех входов имеется возможность подключения внутреннего подтягивающего резистора между входом и шиной питания V_{cc}. В ATmega 128x шесть 8-разрядных порта ввода/вывода (порты A, B, C, D, E, F) и один 5-разрядный порт ввода/вывода G. Всего контактов ввода/вывода 53.

Можно выполнять операции над любым выводом (с помощью команд SBI и CBI), не влияя на другие выводы порта. Это относится к изменению режима работы контакта ввода/вывода, к изменению состояния выходного буфера (для выходов) и к изменению состояния внутреннего подтягивающего резистора (для входов).

Обращение к портам производится через регистры ввода/вывода. Под каждый порт в адресном пространстве ввода/вывода зарезервировано по 3 адреса, по которым размещены следующие регистры: регистр данных порта PORTx, регистр направления данных DDRx и регистр выводов порта PINx. Действительные названия регистров получаются подстановкой названия порта вместо символа «x», соответственно регистры порта A называются PORTA, DDRA, PINA, порта B - PORTB, DDRB, PINB и т. д. Поскольку с помощью регистров PINx осуществляется доступ к физическим значениям сигналов на выводах порта, они доступны только для чтения, тогда как остальные два регистра доступны и для чтения, и для записи.

Каждому выводу порта соответствуют три разряда регистров ввода/вывода да: *PORTxn* (регистр PORTx), DDxn (регистр DDRx) и *PINxn* (регистр PINx). Порядковый номер вывода порта соответствует порядковому номеру восьми, в регистрах порта используется соответствующее число младших разрядов.

Разряд DDxn регистра DDx определяет направление передачи данных через контакт ввода/вывода. Если этот разряд установлен в «1», то n-й вывод порта является выходом, если же сброшен в «0» — входом.

DDRx	7	6	5	4	3	2	1	0
	DDRx.7	DDRx.6	DDRx.5	DDRx.4	DDRx.3	DDRx.2	DDRx.1	DDRx.0
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Разряд *POKTxn* регистра PORTx выполняет двойную функцию. Если вывод функционирует как выход (DDxn = «1»), этот разряд определяет состояние вывода порта. Если разряд установлен в «1», на выводе устанавливается напряжение ВЫСОКОГО уровня. Если разряд сброшен в «0», на выводе Устанавливается напряжение НИЗКОГО уровня.

PORTx	7	6	5	4	3	2	1	0
	PORTx.7	PORTx.6	PORTx.5	PORTx.4	PORTx.3	PORTx.2	PORTx.1	PORTx.0
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Если же вывод функционирует как вход (DDxn = «0»), разряд PORTxn определяет состояние внутреннего подтягивающего резистора для данного вывода. При установке разряда PORTxn в «1» подтягивающий резистор подключается между выводом микроконтроллера и проводом питания.

PINx - из этого регистра можно прочитать текущие логические уровни на выводах МК соответствующего порта

PINx	7	6	5	4	3	2	1	0
	PINx.7	PINx.6	PINx.5	PINx.4	PINx.3	PINx.2	PINx.1	PINx.0
Чтение(R)/Запись(W)	R	R	R	R	R	R	R	R
Начальное значение	0	0	0	0	0	0	0	0

Таблица1 . Влияние регистров DDRx и PORTx на конфигурацию выводов портов.

DDx*n*	Px*n*	Функция вывода	Резистор	Примечания
0	0	вход	отключен	Третье состояние (Hi-Z)
0	1	вход	подключен	При подключении вывода к общему проводу он является источником тока
1	0	выход	отключен	Выход установлен в «0»
1	1	выход	отключен	Выход установлен в «1»

Примечание:
n = 7...0 — номер вывода (разряд порта).

Прерывания

Прерывание прекращает нормальный ход программы для выполнения приоритетной задачи, определяемой внутренним или внешним событием микроконтроллера. При возникновении прерывания микроконтроллер сохраняет в стеке содержимое счетчика команд и загружает в него адрес соответствующего вектора прерывания. По этому адресу, как правило, находится команда безусловного перехода 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.

Для глобального разрешения/запрещения прерываний предназначен флаг I регистра SREG. Для разрешения прерываний он должен быть установлен в 1, а для запрещения — сброшен в 0. При возникновении прерывания флаг I регистра SREG аппаратно сбрасывается, запрещая тем самым обработку следующих прерываний. Однако в подпрограмме обработки прерывания этот флаг можно снова установить в 1 для разрешения вложенных прерываний. При возврате из подпрограммы обработки прерывания (при выполнении команды reti) флаг I устанавливается аппаратно.

SREG	7	6	5	4	3	2	1	0
S3F (S5F)	I	T	H	S	V	N	Z	C
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Микроконтроллеры AVR имеют многоуровневую систему приоритетных прерываний. Младшие адреса памяти программ начиная с адреса $001 отведены под таблицу векторов прерывания. Каждому прерыванию соответствует свой адрес в этой таблице, и именно этот адрес загружается в счетчик команд при возникновении прерывания. Положение вектора в таблице определяет также и приоритет соответствующего прерывания: чем меньше адрес, тем выше приоритет прерывания.

N вектора	Адрес памяти программ(4)	Источник	Условие возникновения прерывания
1	$0000(1)	RESET	Сброс внешний, при подаче или при недопустимом U-питания, сброс сторожевым таймером и через JTAG-интерфейс
2	$0002	INT0	Запрос на внешнее прерывание 0
3	$0004	INT1	Запрос на внешнее прерывание 1
4	$0006	INT2	Запрос на внешнее прерывание 2
5	$0008	INT3	Запрос на внешнее прерывание 3
6	$000A	INT4	Запрос на внешнее прерывание 4
7	$000C	INT5	Запрос на внешнее прерывание 5
8	$000E	INT6	Запрос на внешнее прерывание 6
9	$0010	INT7	Запрос на внешнее прерывание 7
10	$0012	TIMER2_COMP	Совпадение таймера-счетчика 2
11	$0014	TIMER2 OVF	Переполнение таймера-счетчика 2
12	$0016	TIMER1_CAPT	Захват фронта таймером-сч-м 1
13	$0018	TIMER1_COMPA	Совпадение А таймера-счетчика 1
14	$001A	TIMER1_COMPB	Совпадение В таймера-счетчика 1
15	$001C	TIMER1 OVF	Переполнение таймера-счетчика 1
16	$001E	TIMER0_COMP	Совпадение таймера-счетчика 0
17	$0020	TIMER0 OVF	Переполнение таймера-счетчика 0
18	$0022	SPI, STC	Завершение последовательной передачи интерфейсомSPI
19	$0024	USART0, RX	Завершение приема УСАПП 0
20	$0026	USART0,UDRE	Регистр данных УСАПП0 свободен
21	$0028	USART0, TX	Завершение передачи УСАПП 0
22	$002A	ADC	Завершение преобразования АЦП
23	$002C	EE READY	Готовность ЭСППЗУ
24	$002E	ANALOG_COMP	Аналоговый компаратор
25	$0030(3)	TIMER1_COMPC	Совпадение С таймера-счетчика 1
26	$0032(3)	TIMER3_CAPT	Захват фронта таймером счет-м 3
27	$0034(3)	TIMER3_COMPA	Совпадение А таймера-счетчика 3
28	$0036(3)	TIMER3_COMPB	Совпадение В таймера-счетчика 3
29	$0038(3)	TIMER3_COMPC	Совпадение С таймера-счетчика 3
30	$003A(3)	TIMER3 OVF	Переполнение таймера счетчика 3
31	$003C(3)	USART1, RX	Завершение приема УСАПП 1
32	$003E(3)	USART1,UDRE	Рег-р данных УСАПП1 свободен
33	$0040(3)	USART1, TX	Завершение передачи УСАПП1
34	$0042(3)	TWI	Двухпроводной посл-й интерфейс
35	$0044(3)	SPM READY	Готовность записи в память программ

Для разрешения/запрещения внешних прерываний регистр маскирования внешних прерываний EIMSK.

EIMSK	7	6	5	4	3	2	1	0
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Каждый разряд INTx регистра отвечает за разрешение/запрещение прерывания с номером, соответствующим номеру разряда. Если *n* -й разряд регистра установлен в "1" и флаг I регистра SREG также установлен в "1", то прерывание с вывода INT*N* разрешено.

Условия генерации прерываний определяются регистрами EICRA ("A"-регистр управления внешними прерываниями) и EICRB ("B"-регистр управления внешними прерываниями), расположенных по адресам $6A и $3A соответственно.

EICRA	7	6	5	4	3	2	1	0
	ICS31	ICS30	ICS21	ICS20	ICS11	ICS01	ICS01	ICS00
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	0	0	0	0	0	0	0	0

EICRB	7	6	5	4	3	2	1	0
	ICS71	ICS70	ICS61	ICS60	ICS51	ICS50	ICS41	ICS40
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	0	0	0	0	0	0	0	0

ISCn1	ISCn0	Условие
0	0	По НИЗКОМУ уровню на выходе INTn
0	1	зарезервировано
1	0	По спадающему фронту на выходе INTn
1	1	По нарастающему фронту на выходе INTn

Для индикации наступления внешних прерываний в микроконтроллерах Mega используется либо общий регистр флагов прерываний GIFR, либо регистр флагов внешних прерываний EIFR для ATmega128x.

EIFR	7	6	5	4	3	2	1	0
	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Таймеры/счётчики

Счетный регистр таймера/счетчика TCNTn 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 clk_{T0} (clk_{T2}). Независимо от того, присутствует тактовый сигнал или нет, регистр доступен в любой момент времени как для чтения, так и для записи. Однако следует помнить, что любая операция записи в счетный регистр блокирует работу блока сравнения на время одного периода тактового сигнала таймера/счетчика. После подачи напряжения питания в регистре TCNTn находится нулевое значение.

Разрешение прерывания осуществляется установкой в 1 бита регистра маски TIMSK (ETIMSK) (Timer/Counter Interrupt MaSK Register — регистр маски прерываний от таймеров/счетчиков). В модели ATmegal28x этот регистр расположен по адресу $37($57). Разумеется, флаг I регистра SREG также должен быть установлен в 1.

TIMSK	7	6	5	4	3	2	1	0
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

ETIMSK	7	6	5	4	3	2	1	0
	-	-	TICIE3	OCIE3A	OCIE3B	TOIE3	OCIE3C	OCIE1C
Чтение(R)/Запись(W)	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Таблица 2. Разряды регистра TIMSK

Название	Описание
OCIE2	Флаг разрешения прерывания по событию «Совпадение» таймера/счетчика T2
TOIE2	Флаг разрешения прерывания по переполнению таймера/счетчика T2
TICIE1	Флаг разрешения прерывания по событию «Захват» таймера/счетчика T1
OCIE1A	Флаг разрешения прерывания по событию «Совпадение A» гаймера/сче/г
OCIEIB	Флаг разрешения прерывания по событию «Совпадение B» таймера/счетчика T1
TOIE1	Флаг разрешения прерывания по переполнению таймера/счетчика T1
OCIE0	Флаг разрешения прерывания по событию «Совпадение» таймера/счетчикаTO_
TO1E0	Флаг разрешения прерывания по переполнению таймера /счетчика TO

Таблица 3. Разряды регистра ETIMSK

Название	Описание
-	Не используется, читается как «О»
T1C1E3	Флаг разрешения прерывания по событию «Захват» таймера/счетчика Т3
OCIE3A	Флаг разрешения прерывания по событию «Совпадение А» таймера/счетчика Т3
OCIE3B	Флаг разрешения прерывания по событию «Совпадение В» таймера/счетчика Т3
TOIE3	Флаг разрешения прерывания по переполнению таймера/счетчика Т3
OCIE3C	Флаг разрешения прерывания по событию «Совпадение С» таймера/счетчика Т3
OCIEIC	Флаг разрешения прерывания по событию «Совпадение С» таймера/счетчика Т1

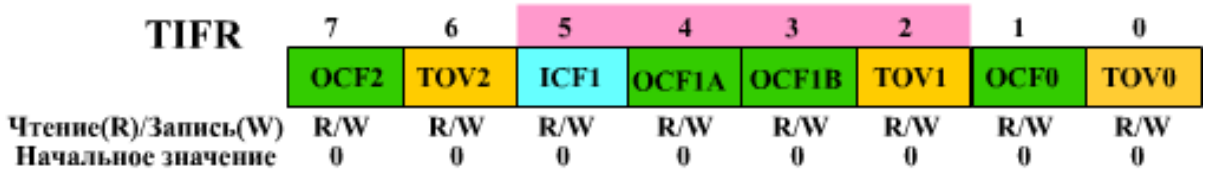

Таблица 4. Разряды регистра TIFR

Название	Описание
OCF2	Флаг прерывания по событию «Совпадение» таймера/счетчика Т2
TOV2	Флаг прерывания по переполнению таймера/счетчика Т2
1CF1	Флаг прерывания по событию «Захват» таймера/счетчика Т1
OCFIA	Флаг прерывания по событию «Совпадение А» таймера/счетчика Т1
OCF1B	Флаг прерывания по событию «Совпадение В» таймера/счетчика Т1
TOV1	Флаг прерывания по переполнению таймера/счетчика Т1
OCF0	Флаг прерывания по событию «Совпадение» таймера/счетчика ТО
TOV0	Флаг прерывания по переполнению таймера /счетчика ТО

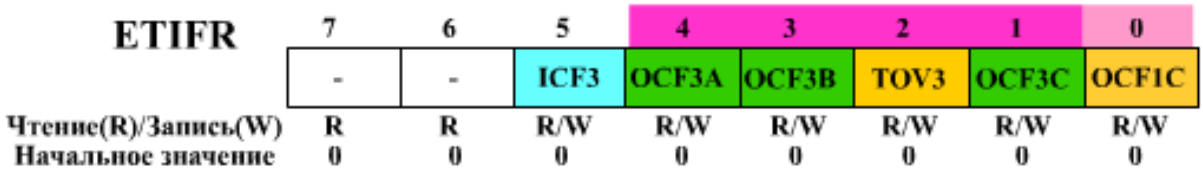

Таблица 5. Разряды регистра ETIFR

Название	Описание
-	Не используется, читается как «0»
ICF3	**Флаг** прерывания по событию «Захват» таймера/счетчика Т3
OCF3A	Флаг прерывания по событию «Совпадение А» таймера/счетчика Т3
OCF3B	Флаг прерывания по событию «Совпадение В» таймера/счетчика Т3
TOV3	Флаг прерывания по переполнению таймера/счетчика Т3
OCF3C	Флаг прерывания по событию «Совпадение С» таймера/счетчика Т3
OCF1C	Флаг прерывания по событию «Совпадение С» таймера/счетчика Т1

Регистры TCCRx предназначены для управления модулем таймера/счетчика. Формат регистров и описание их битов приведены ниже.

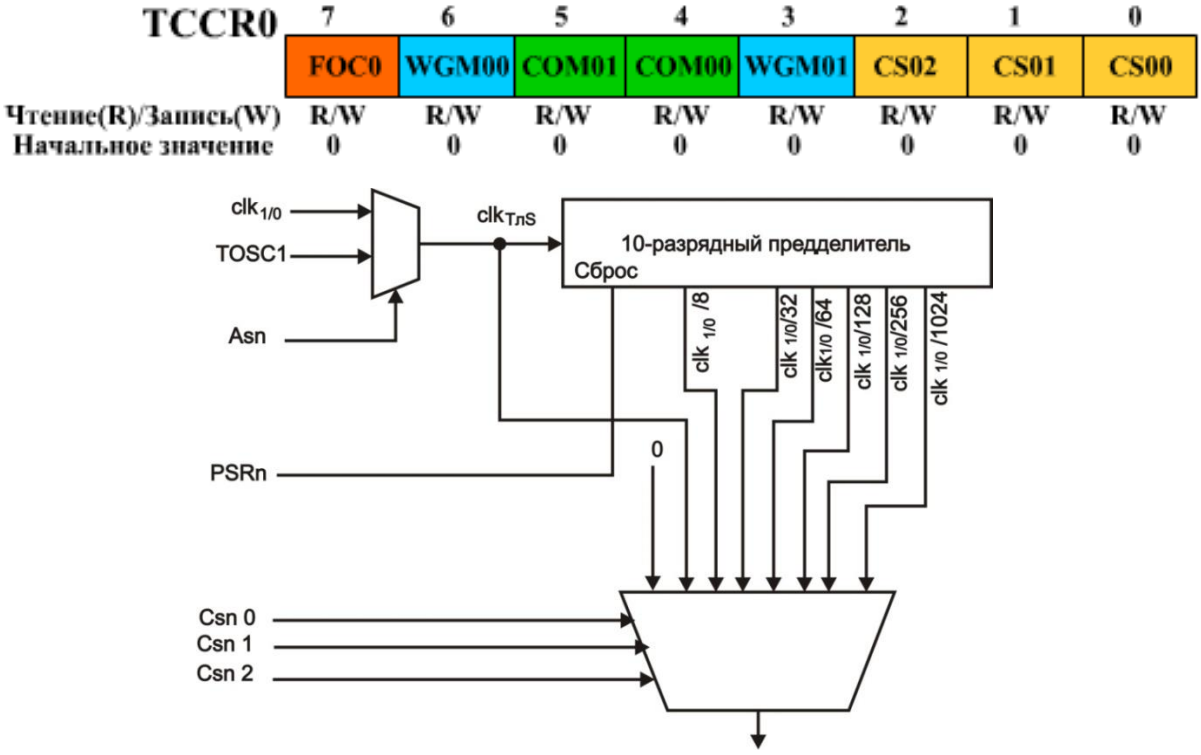

Рис. 1. Блоки предделителей таймеров-счетчиков

CSn2…CSn0 - управление тактовым сигналом. Эти биты определяют источник тактового сигнала таймера/счетчика. Действие этих битов зависит от исполнения таймера/счетчика и описано ниже:

Таблица 6. Управление тактовым сигналом

CSn2	CSn1	CSn0	Источник тактового сигнала		
			Исполнения 1, 2	Исполнение 3	
				ASn = «0»	ASn = «1»
0	0	0	Таймер/счетчик остановлен	Таймер/счетчик остановлен	
0	0	1	clk|1/0	clk|1/0	c1кTOSC1
0	1	0	clki/o/8	<%o/8	c1кTOSC1/8
0	1	1	dki/o/64	clk,/0/32	c1кTOSC1/32
1	0	0	clkI/0/256	clkI/0/64	c1кTOSC1/64
1	0	1	clki/o/1024	c1кI/0/128	c1кTOSC1/128
1	1	0	Вывод Tn,, счет осуществляется по спадающему фронту импульсов	clkI/0/256	c1кTOSC1/256
1	1	1	Вывод Tn, счет осуществляется по нарастающему фронту импульсов	clk,/o/1024	cc1кTOSC1/1024

WGMn1, WGMn0 - режим работы таймера/счетчика. Эти биты определяют режим работы таймера/счетчика следующим образом:

Таблица 7. Выбор режима работы

Номер режима	WGM*n*1	WGM*n*0	Режим работы таймера/счетчика T*n*
0	0	0	Normal
1	0	1	Phase correct PWM
2	1	0	CTC (сброс при совпадении)
3	1	1	Fast PWM

Каждый таймер/счетчик использует один или более выводов микроконтроллера. Как правило, эти выводы — линии портов ввода/вывода общего назначения, а функции, реализуемые этими выводами при работе совместно с таймерами/счетчиками, являются их альтернативными функциями При использовании альтернативных функций линий портов ввода/вывода необходимо, как правило, самостоятельно сконфигурировать эти выводы в соответствии с их функциональным назначением.

COMn1, COMn0 - режим работы блока сравнения. Эти биты определяют поведение вывода OCn при наступлении события «Совпадение». Влияние содержимого этих битов на состояние вывода зависит от режима работы.

Таблица 8. Управление выводом OCXn в режиме Normal

COMn1	COMnO	Описание
0	0	Таймер/счетчик Tn отключен от вывода OCn
0	1	Состояние вывода меняется на противоположное
1	0	Вывод сбрасывается в «0»
1	1	Вывод устанавливается в «1»

FOCn - принудительное изменение состояния вывода OC*n* (режимы Normal и CTC). При записи лог. 1 в этот бит состояние вывода OC*n* изменяется в соответствии с установками битов COM0 и COM1. Прерывание при этом не генерируется и сброс таймера (в режиме CTC) не производится. В режимах Fast PWM и Phase Correct PWM этот бит должен быть сброшен в 0. При чтении бита всегда возвращается 0.

Режим Normal

В режиме Normal счетный регистр функционирует как обычный суммирующий счетчик. По каждому импульсу тактового сигнала clk_T осуществляется инкрементирование счетного регистра. При переходе через значение $FF возникает переполнение, и счет продолжается со значения $00. В том же такте сигнала clk_T в котором обнуляется регистр TCNTn, флаг прерывания по переполнению TOV*n* устанавливается в 1.

При равенстве счетного регистра и регистра сравнения устанавливается соответствующий флаг прерывания OCF*n* (OCF*n*A/OCF*n*B) и, если бит OCIE*n* (OCIE*n*A/ OCIE*n*B) регистра маски установлен в 1, генерируется прерывание. Наряду с установкой флага при равенстве счетного регистра и регистра сравнения может изменяться состояние вывода OC*n* (OC*n*A/OC*n*B) микроконтроллера.

Режим CTC (сброс при совпадении).

В этом режиме счетный регистр тоже функционирует как обычный суммирующий счетчик, инкрементирование которого осуществляется по каждому импульсу тактового сигнала clk_T. Однако максимально возможное значение счетного регистра и, следовательно, разрешающая способность счетчика определяются регистром сравнения OCRn (OCRnA). После достижения значения, записанного в регистре сравнения, счет продолжается со значения $00. Если в регистре сравнения записано $FF, то в том же такте сигнала clk_T, в котором обнуляется счетный регистр, устанавливается флаг прерывания по переполнению TOV*n* в соответствующем регистре флагов.

При достижении счетчиком максимального значения устанавливается флаг OCF*n* (OCF*n*A), и, если бит OC1E*n* (OC1E*n*A) соответствующего регистра маски установлен в 1, генерируется прерывание. Одновременно с установкой флага может изменяться состояние выводов OC*n* (OC*n*A/ OC*n*B) микроконтроллера. Состояние выводов определяется битами COMn1; COMлn регистра управления TCCR*n*/ TCCR*n*A.

Режим Fast PWM.

Режим Fast PWM («Быстродействующий ШИМ») позволяет генерировать высокочастотный сигнал с широтно-импульсной модуляцией. В связи с высокой частотой генерируемого сигнала данный режим с успехом может использоваться в таких приложениях, как регулирование мощности, выпрямление, цифро-аналоговое преобразование и др.

Счетный регистр в этом режиме функционирует как суммирующий счетчик, инкрементирование которого осуществляется по каждому импульсу тактового сигнала clk_T. Состояние счетчика изменяется от \$00 до максимального значения, после чего счетный регистр сбрасывается и цикл повторяется. При достижении счетчиком максимального значения устанавливается флаг прерывания по переполнению TOV*n* в соответствующем регистре флагов, а при равенстве содержимого счетного регистра и регистра сравнения OCR*n* устанавливается флаг OCF*n*. Максимальное значение равно \$FF (при WFMn2 = 0 или при отсутствии этого бита в регистре микроконтроллера) или задается регистром OCRnA (при WFMn2 = 1).

Особенностью работы схемы сравнения в этом режиме является двойная буферизация записи в регистр OCR*n*, которая заключается в том, что записываемое число на самом деле сохраняется в специальном буферном регистр, а изменение содержимого регистра сравнения происходит только в момент достижения счетчиком максимального значения. Благодаря такому решению исключается появление несимметричных импульсов сигнала (помех) на выходе модулятора, которые были бы неизбежны при непосредственной записи в регистр сравнения.

Режим Phase Correct PWM

Режим Phase Correct PWM («ШИМ с точной фазой»), как и режим Fast PWM, предназначен для генерации сигналов с широтно-импульсной модуляцией. Однако в этом режиме счетный регистр функционирует как реверсивный счетчик, изменение состояния которого осуществляется по каждому импульсу тактового сигнала clk_{T0} (clk_{T2}). Состояние счетчика сначала изменяется от \$00 до максимального значения, а затем обратно до \$00. Соответственно, максимальная частота сигнала в этом режиме в 2 раза меньше

максимальной частоты сигнала в режиме Fast PWM. Тем не менее благодаря «симметричности» изменения состояния счетчика режим Phase Correct PWM предпочтительнее использовать для решения задач управления двигателями.

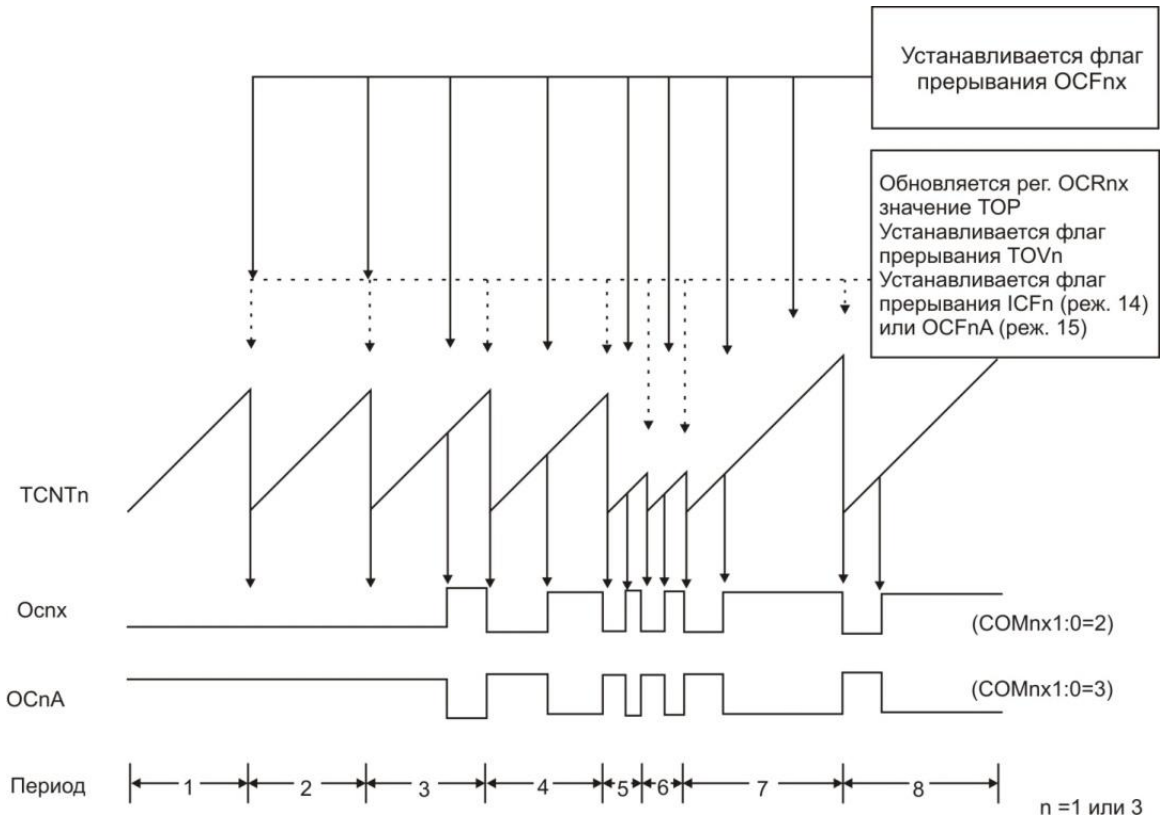

Рис. 2. Формирование ШИМ-сигнала в режиме Fast PWM

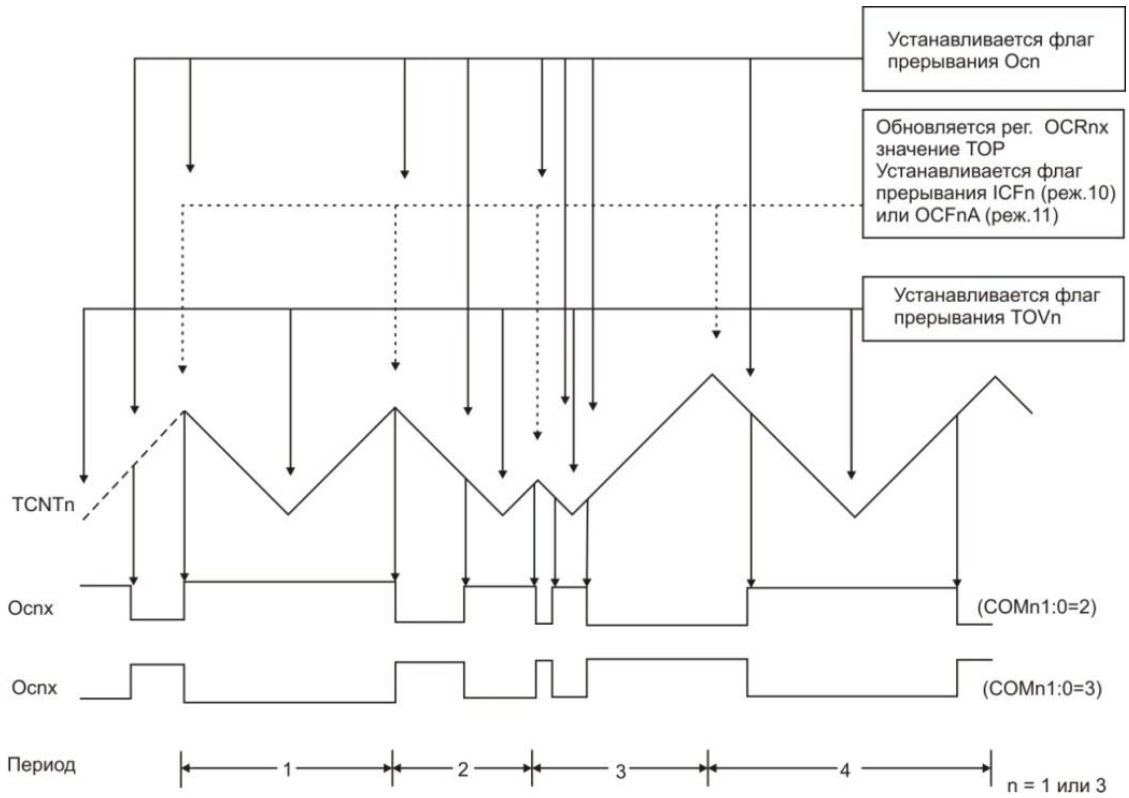

Рис. 3. Формирование ШИМ-сигнала в режиме Phase Correct PWM

16разрядные таймеры/счетчики

Для управления таймером/счетчиком используются три регистра управления: TCCR1A (TCCR3A), TCCR1B (TCCR3B), TCCR1C (TCCR3C). Формат регистров TCCR1A *(a)* и TCCR3A *(6)*

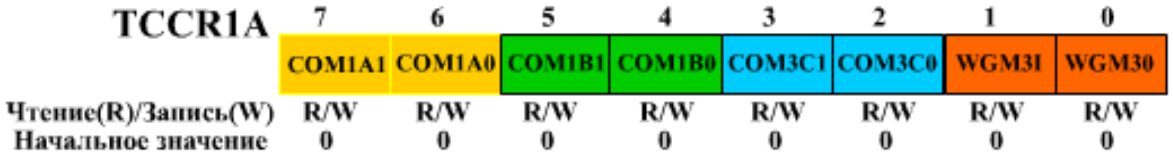

Таблица 9. Разряды регистра TCCR1A (TCCR3A)

Разряды	Название	Описание
7,6	COMnA1:COMnA0	Режим работы блока сравнения *x.* Эти разряды определяют поведение вывода *OCnx* при наступлении события «Совпадение».
5,4	COMnB1:COMnB0	
3,2	COMnC1:COMnC0	
1,2	WGMnl:WGMnO	Режим работы таймера/счетчика. Совместно с разрядами WGMn3:WGMn2 регистра TCCRnB определяют режим работы таймера/счетчика Tn

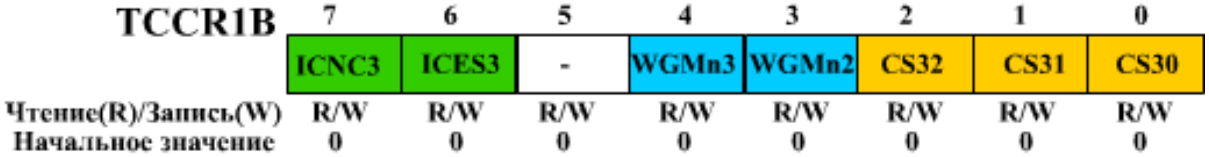

Таблица 10. Разряды регистра TCCR1B (TCCR3B)

Разряд	Название	Описание
7	ICNCn	Управление схемой подавления помех блока захвата. Если разряд сброшен в «О», схема подавления помех выключена (захват] производится по первому активному фронту). Если установлен в «1», схема подавления помех включена и захват осуществляется только в случае 4-х одинаковых выборок, соответствующих фронту сигнала
6	ICESn	Выбор активного фронта сигнала захвата. Если разряд ICESn сброшен в «0», сохранение счетного регистра в регистре захвата осуществляется по спадающему фронту сигнала. Если установлен в «1». сохранение счетного регистра в регистре захвата осуществляется по нарастающему фронту сигнала. Одновременно устанавливается флаг прерывания lCFn регистра T1FR (ET1FR)
5	-	Не используется, читается как «0»
4,3	WGMn3: WGMn2	Режим работы таймера/счетчика. Совместно с разрядами WGMnl:WGMnO регистра TCCRnO определяют режим работы таймера/счетчика Tn
2...0	CSn2...CS n0	Управление тактовым сигналом. Эти разряды определяют источник тактового сигнала микроконтроллера

TCCR1C	7	6	5	4	3	2	1	0
	ICNC3	ICES3	-	WGMn3	WGMn2	CS32	CS31	CS30
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Таблица 11. Разряды регистра TCCR1C (TCCR3C)

Разряд	Название	Описание
7	FOCnA	Принудительное изменение состояния вывода OCnx При записи в разряд - FOCnA- лог. 1 состояние вывода OCnx изменяется в соответствии с установками разрядов COMn1x:COMn0x регистра TCCRnA. Прерывание при этом генерируется и сброс таймера (в режиме CTC) не производится. Эта функция доступна только в тех режимах, которые не используются для генерации сигнала с ШИМ.
6	FOCnB	
5	FOCnC	
4...0	-	Не используется, читается как «0»

Режим работы таймера/счетчика Т1 (Т3) определяется состоя разрядов WGMw3:WGMw0. Зависимость режима работы таймеров/счетчиков от состояния этих разрядов показана в **Табл**

Таблица 12. Режимы работы таймеров/счетчиков Т1 и Т3

WGMn3	WGMn2 (CTC1)	WGMnl (PWM11)	WGMnO (PWM10)	Режим работы тай мера/счетчика Тл	Модуль счета (TOP)
0	0	0	0	Normal	SFFFF
0	0	0	1	Phase correct PWM, 8-разрядный	SOOFF
0	0	1	0	Phase correct PWM^ 9-разрядный	S01FF
0	0	1	1	Phase correct PWM, 10-разрядный	S03FF
0	1	0	0	CTC (сброс при совпадении)	OCR/iA
0	1	0	1	Fast PWM, 8-разрядный	SOOFF
0	1	1	0	Fast PWM, 9-разрядный	$01FF
0	1	1	1	Fast PWM, 10-разрядный	S03FF
1	0	0	0	Phase and Frequency Correct PWM	ICR/i
1	0	0	1	Phase and Frequency Correct PWM	OCR/iA
1	0	1	0	Phase correct PWM	ICR/i
1	0	1	1	Phase correct PWM	DCR/JA
1	1	0	0	CTC (сброс при совпадении)	CR/i
1	1	0	1	Зарезервировано	-
1	1	1	0	Fast PWM	ICR/i
1	1	1	1	FastPWM	ОСклА

Обращение к 16-разрядным регистрам

Каждый 16-разрядный регистр таймеров/счетчиков физически размещается в двух 8-разрядных регистрах. Соответственно для обращения к ним требуется выполнить по две операции чтения или записи. Для того чтобы запись или чтение обоих байт содержимого 16-разрядного регистра происходила одновременно, в составе каждого таймера/счетчика имеется специальный 8-разрядный регистр TEMP, предназначенный для хранения старшего байта значения (этот регистр программно недоступен).

Для выполнения цикла записи 16-разрядного регистра первым должен быть загружен старший байт, который помещается в регистр TEMP. При последующей записи младшего байта он объединяется с содержимым регистра TEM P, и оба байта одновременно записываются в 16-разрядный регистр. Если требуется изменить несколько 16-разрядных регистров таймера/счетчика, а старшие байты всех записываемых значений одинаковы, загрузку старшего байта достаточно выполнить только один раз.

Для выполнения цикла чтения 16-разрядного регистра первым должен быть прочитан младший байт. При его чтении содержимое старшего байта помещается в регистр TEMP. При последующем чтении старшего байта возвращается значение, сохраненное в регистре TEMP.

При выполнении цикла обращения к 16-разрядному регистру таймера/счетчика прерывания должны быть запрещены.

Асинхронный режим

В модели ATmegal28x в асинхронном режиме может работать таймер/счетчик T0. В асинхронном режиме на вход предделителя поступает сигнал от кварцевого генератора таймера/счетчика, что позволяет использовать таймер/счетчик в качестве часов реального времени. Задатчиком частоты сигнала может быть как кварцевый резонатор, подключаемый к вывода» TOSC1 и TOSC2 микроконтроллера, так и сигнал от внешней схемы, подаваемый на вывод TOSC1. Несмотря на то что тактовый генератор таймера/счетчика настроен на частоту 32768 Гц, частота кварцевого резонатор либо сигнала от

внешней схемы может лежать в пределах 0...256 кГц. При этом она должна быть в четыре раза меньше частоты тактового сигнала микроконтроллера.

Непосредственная запись в регистры TCNT0 (TCNT2), OCR0 (OCR2) TCCR0 (TCCR2) в асинхронном режиме синхронизируется с тактовым сигналом таймера/счетчика. При записи числа в любой из указанных регистров оно сохраняется в специальном временном регистре, своем для каждого регистра таймера/счетчика. А пересылка содержимого временного регистра в рабочий регистр таймера/счетчика осуществляется по третьему после записи положительному фронту сигнала на выводе TOSC1. Соответственно запись нового значения можно производить только после пересылки содержимого временного регистра в регистр таймера/счетчика.

Для определения момента действительного изменения регистров таймера/счетчика, а также для переключения таймера/счетчика в асинхронный режим предназначен регистр ASSR. Формат регистра ASSR:

-	-	-	-	ASn	TCNnUB	OCRnUB	TCRnUB

Таблица 13. Разряды регистра состояния асинхронного режима ASSR

-	Зарезервированы, читаются как «0»
ASn	Переключение режима работы. Если разряд установлен в "1" на вход предделитепя таймера/счетчика Тл поступают импульсы с кварцевого генератора таймера/счетчика (асинхронный режим). Если разряд сброшен в «0», на вход предделителя поступает внутренний тактовый сигнал микроконтроллера. При изменении состояния этого разряда содержимое регистров TCNT2, OCR2 и TCCR2 может быть повреждено
TCNnUB	Состояние **обновления регистра TCNTn.** При записи в регистр TCNT/n этот флаг устанавливается в «1», а после пересылки записываемого значения в этот регистр, флаг аппаратно сбрасывается в «0». Таким образом, сброшенный флаг TCNnUB означает, что регистр TCNTn готов для записи в него нового значения.
OCRnUB	Состояние обновления регистра OCRn. При записи в регистр OCRn этот флаг устанавливается в «I», а после пересылки записываемого значения в этот регистр флаг аппаратно сбрасывается в «0». Таким образом, сброшенный флаг OCRnUB означает, что регистр OCRn готов для записи в него нового значения.
TCRnUB	Состояние **обновления** регистра TCCRn. При записи **в регистр** TCCRn этот флаг устанавливается в «I», а после пересылки записываемого значения ния в этот регистр флаг аппаратно сбрасывается в «О». Таким образом, сброшенный флаг TCRnUB означает, что регистр TCCRn готов для записи си в него нового значения.

Аналоговый компаратор

Модуль аналогового компаратора позволяет сравнивать значения напряжений, присутствующих на двух выводах микроконтроллера. Результатом сравнения является логическое значение, которое может быть прочитано из программы. По результату сравнения может быть сгенерировано прерывание, а также осуществлен захват состояния таймера/счетчика T1. Последняя функция позволяет, в частности, измерять длительности аналоговых сигналов. Используемые компаратором выводы являются контактами портов ввода/вывода общего назначения.

Таблица 14. Выводы, используемые аналоговым компаратором

Название	ATMega128x	Назначение
AIN0	PE2	Не инвертирующий вход
AIN1	PE3	Инвертирующий вход

Чтобы указанные выводы могли использоваться аналоговым компаратором, они должны быть сконфигурированы как входы (соответствующий разряд регистра DDRx установлен в «0»). Кроме того, необходимо отключить внутренние подтягивающие резисторы записью лог. 0 в соответствующий разряд регистра **PORTx.** Управление компаратором и контроль его состояния осуществляется с помощью регистра ACSR, который во всех моделях расположен по адресу $08 ($28). Формат регистра ACSR:

	7	6	5	4	3	2	1	0
ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Таблица 15. Разряды регистра ACSR

Название	Описание
ACD	Выключение компаратора («О» — включен,«1» - выключен
ACBG AINBG*	Подключение к неинвертирующему входу компаратора внутреннего ИОН («О» - ие подключен, «1» - подключен
ACO	Результат сравнения (выход компаратора)
ACI	Флаг прерывания от компаратора
ACIE	Разрешение прерывания от компаратора
ACIC	Подключение компаратора к схеме захвата таймера/счетчика TI («1» — подключен, «0» — отключен)
ACIS1:ACISO	Условие возникновения прерывания от компаратору

По своему действию рассматриваемый узел микроконтроллера является обычным компаратором. Если напряжение на выводе AIN0 (неинвертируюший вход) больше напряжения на выводе AINI (инвертирующий вход), то результат сравнения будет равен «I». В противном случае результат сравнения будет равен «0». Этот результат (состояние выхода компаратора) сохраняется в разряде ACO регистра ACSR.

Разряд ACD отвечает за включение и выключение компаратора. Поскольку при подаче напряжения питания все разряды регистра ACSR сбрасываются в «0», компаратор включается автоматически при включении микроконтроллера.

Таблица 16. 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 комларатора с «0» на **«1»**

Помимо генерации прерывания, компаратор также может управлять схемой захвата таймера/счетчика T1. Для этого необходимо установить в "1" разряд ACIC регистра ACSR. В результате выход компаратора подключится' к схеме захвата вместо вывода ICP1 микроконтроллера. Если же разряд AC - 1C сброшен в «0», компаратор полностью отключен от блока захвата.

Компаратор может сравнивать сигналы, присутствующие не только на выводах AIN0 и A1N1. Так, вместо вывода AIN0 микроконтроллера к неинвертирующему входу компаратора может быть подключен внутренний источник опорного напряжения (ИОН) величиной 1.22 ±0.1 В. Для этого необходимо установить в «1» разряд ACBG регистра ACSR.

На инвертирующий вход компаратора может также поступать сигнал с выхода мультиплексора модуля АЦП. Другими словами, вместо вывода AIN1 микроконтроллера инвертирующий вход компаратора может быть подключен к любому из входов АЦП ADC0...ADC7 .

Подключение выхода мультиплексора АЦП к входу компаратора осуществляется установкой в «1» разряда ACME регистра специальных

функций SFIOR (3-й разряд регистра). Разумеется, модуль АЦП при этом должен быть выключен (разряд ADEN регистра ADCSRA сброшен в «О»). Какой именно из входов АЦП будет использоваться в качестве инвертирующего входа компаратора, определяется разрядами MUX2...0 регистра ADMUX, как показано в Табл. 2.110.

Таблица 17. Управление инвертирующим входом компаратора

ACME	ADEN	MUX2...0	Инвертирующий вход компаратора
0	X	XXX	AIN1
1	1	XXX	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	o	100	ADC4
1	0	101	ADC5
1	0	110	ADC6*
1	0	111	ADC7*

Аналого-цифровой преобразователь.

Модуль 10-битного АЦП последовательного приближения. Основные параметры этого АЦП следующие: абсолютная погрешность: ±2 LSB, интегральная нелинейность: ±0.5 LSB, быстродействие: до 15 тыс. выборок/с.

На входе модуля АЦП имеется 8-канальный аналоговый мультиплексор, предоставляющий в распоряжение пользователя 8 (16) каналов с несим-метричными входами. Входы АЦП могут объединяться попарно для формирования различного числа каналов с дифференциальным входом. При этом в некоторых каналах имеется возможность 10- и 200-кратного предварительного усиления входного сигнала. При коэффициентах усиления 1х и 10х действительная разрешающая способность АЦП по этим каналам составляет 8 бит, а при коэффициенте усиления 200х — 7 бит. В качестве источника опорного напряжения для АЦП может использоваться как напряжение питания микроконтроллера, так и внутренний либо внешний источник опорного напряжения.

Формат регистра ASCSR:

	7	6	5	4	3	2	1	0
ASCSRA	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Таблица 18. Разряды регистра ASCSRA

Бит	Название	Описание
7	ADEN	Разрешение работы АЦП (1 — включено, 0 — выключено)
6	ADSC	Запуск преобразования (1 — начать преобразование)
5	ADATE	Выбор режима работы АЦП
4	ADIF	Флаг прерывания от компаратора
3	ADIE	Разрешение прерывания от компаратора
2...0	ADPS2:ADPS0	Выбор частоты преобразования

Режим работы определяется состоянием разряда ADFR. Если он установлен в 1, АЦП работает в режиме непрерывного преобразования. В этом режиме запуск каждого следующего преобразования осуществляется автоматически после окончания предыдущего. Если же разряд сброшен в 0, АЦП работает в режиме одиночного преобразования и запуск каждого преобразования осуществляется по команде пользователя.

Для формирования тактовой частоты модуля АЦП в нем имеется отдельный предделитель. Наибольшая точность преобразования достигается, если тактовая частота модуля АЦП находится в диапазоне 50...200 кГц. Соответственно, коэффициент деления предделителя рекомендуется выбирать таким, чтобы тактовая частота модуля АЦП находилась в указанном диапазоне. Если же точности преобразования меньше 10 битов достаточно, можно использовать более высокую частоту, увеличивая тем самым частоту выборки.

Таблица 19. Задание коэффициента деления АЦП

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

Во всех моделях, кроме ATmega8x и ATmegal28x, запуск АЦП возможен не только по команде пользователя, но и по прерыванию от некоторых пе-

риферийных устройств, имеющихся в составе микроконтроллера. Для выбора режима работы в этих моделях используется бит ADATE регистра ASCSR и биты ADTS2… ADTS0 регистра ADCSRB.

	7	6	5	4	3	2	1	0
ADCSRB	-	-	-	-	-	ADTS2	ADTS1	ADTS0
Чтение(R)/Запись(W)	R	R	R	R	R	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Таблица 20. Задание источника стартового сигнала

ADTS2	ADTS1	ADTS0	Источник стартового сигнала
0	0	0	Режим непрерывного преобразования
0	0	1	Прерывание от аналогового компаратора
0	1	0	Внешнее прерывание INT0
0	1	1	Прерывание по событию «Совпадение» («Совпадение А») таймера/счетчика T0
1	0	0	Прерывание по переполнению таймера/счетчика T0
1	0	1	Прерывание по событию «Совпадение B» таймера/счетчика T1
1	1	0	Прерывание по переполнению таймера/счетчика T1
1	1	1	Прерывание по событию «Захват» таймера/счетчика T1

Регистр управления мультиплексором АЦП-

ADMUX:

	7	6	5	4	3	2	1	0
ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
Чтение(R)/Запись(W)	R	R	R	R	R	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Таблица 21. Биты регистра ADMUX

Бит	Название	Описание
7,6	REFS1-.REFS0	Выбор источника опорного напряжения
5	ADLAR	Выравнивание результата преобразования
4…0	MUX4...MUX0	Выбор входного канала

Таблица 22. Выбор источника опорного напряжения

REFS1	REFS0	Описание
0	0	Внешний источник опорного напряжения
0	1	Напряжение питания AVcc
1	1	Внутренний ИОН 2,56 В на выводе AREF

По завершении преобразования (ADIF = 1) результат может быть считан из пары регистров результата преобразования АЦП (ADCL, ADCH).

Интерфейс USART (универсальный синхронный/асинхронный приемо-передатчик).

Все модули приемопередатчиков обеспечивают полнодуплексный обмен по последовательному каналу, при этом скорость передачи данных может варьироваться в довольно широких пределах. Длина посылки может составлять от 5 до 9 битов. Во всех модулях в обязательном порядке присутствуют схемы контроля и формирования бита четности.

Модули USART, реализованные в микроконтроллерах семейства, могут обнаруживать следующие внештатные ситуации: переполнение, ошибка кадрирования, неверный старт-бит.

Предусмотрены три прерывания, запрос на генерацию которых формируется при наступлении следующих событий: «передача завершена», «регистр данных передатчика пуст» и «прием завершен».

Буферные регистры приемника и передатчика располагаются по одному адресу пространства ввода/вывода и обозначаются как регистр данных UDR (UDR*n*). В этом регистре хранятся младшие 8 битов принимаемых и передаваемых данных. При чтении регистра UDR выполняется обращение к буферному регистру приемника, при записи — передатчика.

Для управления модулями USART используются три регистра: UCSRA (UCSR*n*), UCSRB (UCSR*n*B) и UCSRC (UCSR*n*C).

Формат регистров UCSRA (UCSR*n*), UCSRB (UCSR*n*B) и UCSRC (UCSR*n*C) приведен на ниже, а значение битов этих регистров описано в таблице 19-21 соответственно.

UCSRnA	7	6	5	4	3	2	1	0
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn
Чтение(R)/Запись(W)	R	R/W	R	R	R	R	R/W	R/W
Начальное значение	0	0	1	0	0	0	0	0

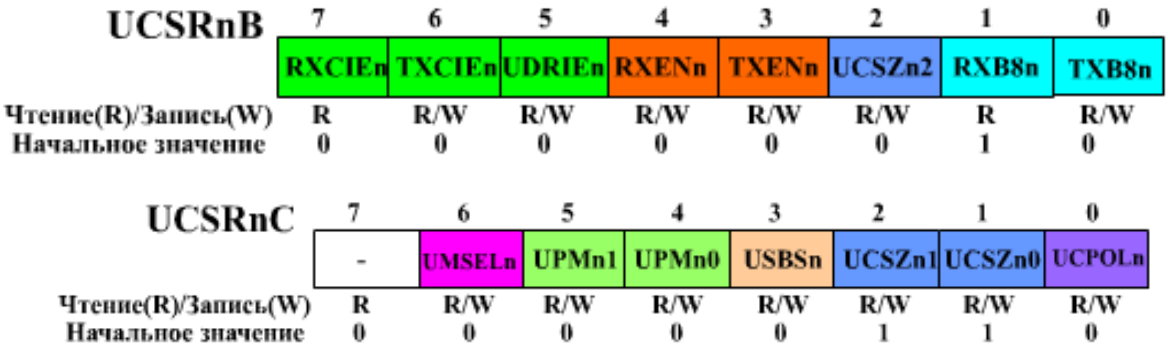

Таблица 23. Биты регистров UCSRA (UCSR*n*).

Название	Описание
RXC	**Флаг завершения приема.** Флаг устанавливается в 1 при наличии непрочитанных данных в буфере приемника (регистр данных UDR). Сбрасывается флаг аппаратно после опустошения буфера.
TXC	**Флаг завершения передачи.** Флаг устанавливается в 1 после передачи всех битов посылки из сдвигового регистра передатчика при условии, что в регистр данных UDR не было загружено новое значение. Флаг сбрасывается аппаратно при выполнении подпрограммы обработки прерывания или программно, записью в него лог. **1**
UDRE	**Флаг опустошения регистра данных.** Данный флаг устанавливается в 1 при пустом буфере передатчика (после пересылки байта из регистра данных UDR в сдвиговый регистр передатчика). Установленный флаг означает, что в регистр данных можно загружать новое значение. Если бит UDRIE регистра UCSRB установлен, генерируется запрос на прерывание «регистр данных пуст». Флаг сбрасывается аппаратно, при записи в регистр данных
FE	Флаг **ошибки кадрирования.** Флаг устанавливается в 1 при обнаружении ошибки кадрирования, т. е. если первый стоп-бит принятой посылки равен 0. Флаг сбрасывается при приеме стоп-бита, равного **1**
DOR	**Флаг переполнения.** Флаг устанавливается в 1, если в момент обнаружения нового старт-бита в сдвиговом регистре приемника находится последнее принятое слово, а буфер приемника полон (содержит два байта). Флаг сбрасывается при пересылке принятых данных из сдвигового регистра приемника в буфер
UPE	**Флаг ошибки контроля четности.** Флаг устанавливается в 1, если в данных, находящихся в буфере приемника, выявлена ошибка контроля четности. При отключенном контроле четности этот бит постоянно сброшен в 0
U2X	**Удвоение скорости обмена**. Если этот бит установлен в 1, то коэффициент деления предделителя контроллера скорости передачи уменьшается с 16 до 8, удваивая тем самым скорость асинхронного обмена по последовательному каналу. Этот бит используется только при асинхронном режиме работы и в

	синхронном режиме должен быть сброшен
MPCM	**Режим мультипроцессорного обмена**. Если этот бит установлен в 1, ведомый микроконтроллер ожидает приема кадра, содержащего адрес. Кадры, не содержащие адреса устройства, игнорируются

Таблица 24. Биты регистров UCSRB (UCSR*n*B).

Название	Описание
RXC1E	**Разрешение прерывания по завершении приема**. Если данный бит установлен в 1, то при установке флага RXC регистра UCSRA генерируется прерывание «прием завершен» (если флаг I регистра SREG установлен в1)
TXCIE	**Разрешение прерывания по завершении передачи**. Если данный бит установлен в 1, то при установке флага TXC регистра UCSRA генерируется прерывание «передача завершена
UDR1E	**Разрешение прерывания при очистке регистра данных UART**. Если данный бит установлен в 1, то при установке флага UDRE регистра UCSRA генерируется прерывание «регистр данных пуст» (если флаг I регистра SREG установлен в 1)
RXEN	**Разрешение приема**. При установке этого бита в 1 разрешается работа приемника USART и переопределяется функционирование вывода RXD. При сбросе бита RXEN работа приемника запрещается, а его буфер сбрасывается. Значения флагов TXC, DOR и FE при этом становятся недействительными
TXEN	**Разрешение передачи**. При установке этого бита в 1 разрешается работа пере датчика UART и переопределяется функционирование вывода TXD Если бит сбрасывается в 0 во время передачи, то выключение передатчика произойдет только после завершения передачи данных, находящихся в сдвиговом регистре и буфере передатчика
UCSZ2	**Формат посылок**. Этот бит совместно с битами UCSZ1:0 регистра UCSRC используется для задания размера слов данных, передаваемых по последовательному каналу
RXB8	**8-й бит принимаемых данных**. При использовании 9-битиых слов данных этот бит содержит значение старшего бита принятого слова. Содержимое этого бита должно быть считано до прочтения регистра данных UDR
TXB8	**8-й бит передаваемых данных.** При использовании 9-битных слов данных содержимое этого бита является старшим битом передаваемого слова. Требуемое значение должно быть занесено в этот бит до загрузки байта данных в регистр UDR

Таблица 25. Биты регистров UCSRC (UCSR*n*C).

Название	Описание
URSEL	**Выбор регистра**. Этот бит определяет, в какой из регистров модуля производится запись. Если бит установлен в 1, обращение производится к регистру UCSRC. Если сброшен в 0, обращение производится к регистру UBRRH.
UMSEL	**Режим работы USART.** Если бит сброшен в 0, то модуль работает в асинхронном режиме. Если бит установлен в 1, то в синхронном
UPM1	**Режим работы схемы контроля и формирования бита четности**. Эти биты определяют функционирование схем контроля и формирования бита четности
UPM0	
USBS	**Количество стоп-битов.** Бит определяет количество стоп-битов. Если бит сброшен в 0, передатчик посылает 1 стоп-бит, если установлен в 1, то 2.
UCSZ1	**Формат посылок**. Совместно с битом UCSZ2 регистра UCSRB эти биты определяют количество битов данных в посылках (размер слова)
UCSZ0	

UCPOL	Полярность тактового сигнала. Определяет момент выдачи и считывания данных на выводах модуля. Бит используется только при работе в синхронном режиме. При работе в асинхронном режиме он должен быть сброшен в 0.		
	UCPOL	**Выдача** данных на вывод TXD	Считывание данных с вывода RXD
	0	Спадающий фронт XCK	Нарастающий фронт XCK
	1	Нарастающий фронт XCK	Спадающий фронт XCK

Под кадром будем понимать совокупность одного слова данных и сопутствующей информации. Кадр начинается со старт-бита, за которым следует младший бит слова данных. После старшего бита слова данных следует один или два стоп-бита. Если включена схема формирования бита четности, он включается между старшим битом слова данных и первым стоп-битом.

Формат кадра определяется различными битами регистров UCSRB (UCSR*n*B) и UCSRC (UCSR*n*C). В частности, размер слова данных определяется битами UCSZ2...UCSZ0 (UCSZ*n*2...UCSZ*n*O) в соответствии с табл.

Таблица 26. Определение размера слова данных

UCSZ2	**UCSZ1**	**UCSZ0**	**Размер слова данных**
0	0	0	5 бит
0	0	1	6 бит
0	1	0	7 бит
0	1	1	8 бит
1	1	1	9 бит

Выбор количества стоп-битов осуществляется с помощью бита USBS (USBS*n*) регистра UCSRC (UCSR*n*C). Если этот бит сброшен в 0, блок передатчика формирует 1 стоп-бит в конце посылки. В противном случае, если бит установлен в 1, блок передатчика формирует 2 стоп-бита. Следует отметить, что приемником второй стоп-бит игнорируется, и соответственно ошибки кадрирования выявляются только для первого стоп-бита.

Биты UPM1…UPM0 (UPM*n*1:UPM*n*0) регистра UCSRC (UCSR*n*C) определяют функционирование схемы контроля четности модулей USART согласно таблице. 23.

Таблица 27. Функционирование схемы контроля четности

UPM1	**UPM0**	**Режим работы схемы**
0	0	Выключена
0	1	Зарезервировано
1	0	Включена, проверка на четность(even parity)
1	1	Включена, проверка на нечетность (odd parity)

Скорость приема/передачи

В асинхронном режиме, а также в синхронном режиме при работе в качестве ведущего скорость приема и передачи данных задается контроллером скорости передачи, работающим как делитель системного тактового сигнала с программируемым коэффициентом деления. Коэффициент определяется содержимым регистра контроллера UBRR. В блок приемника сформированный сигнал поступает напрямую, а в блок передатчика — через дополнительный делитель, коэффициент деления которого (2, 8 или 16) зависит от режима работы модуля USART.

Скорость обмена определяется следующим образом:

- асинхронный режим (обычный, U2X*n* = 0)

$$BAUD = \frac{f_{CK}}{16 \cdot (UBRR + 1)}$$

- асинхронный режим (ускоренный, U2X*n* = 1)

$$BAUD = \frac{f_{CK}}{8 \cdot (UBRR + 1)}$$

- синхронный режим ведущего

$$BAUD = \frac{f_{CK}}{2 \cdot (UBRR + 1)},$$

BAUD — скорость передачи, бит/с;
f_{CK} — тактовая частота микроконтроллера;
UBRR — содержимое регистра контроллера скорости передачи (0...4095).

Интерфейс SPI

Последовательный периферийный интерфейс SPI (Serial Peripheral Interface имеет два назначения. Во-первых, с его помощью может осуществляться обмен данными между микроконтроллером и различными периферийными устройствами. Посредством этого интерфейса также может производиться обмен данными между несколькими микроконтроллерами AVR. Кроме того, через интерфейс SPI может быть осуществлено программирование микроконтроллера. При обмене данными по интерфейсу SPI микроконтроллер AVR может работать как ведущий (режим Master) либо как ведомый (режим

Slave). Модуль SPI использует четыре вывода микроконтроллера.

Для управления модулем SPI предназначен регистр управления SPCR. В модели ATmega128x этот регистр располагается по адресу $0D ($2D). Форматы регистров SPCR и SPSR:

SPCR	7	6	5	4	3	2	1	0
	SPIE	SPE	DOR0	MSTR	CPOL	CPHA	SPR1	SPR0
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Таблица 28. Разряды регистра SPCR

Бит	Название	Описание
7	SPIE	Разрешение прерывания от SPI
6	SPE	Включение/выключение SPI
5	DORD	Порядок передачи данных
4	MSTR	Выбор режима работы (Master/Slave)
3	CPOL	Полярность тактового сигнала
2	CPHA	Фаза тактового сигнала
1, 0	SPR1:SPR0	Скорость передачи

Контроль состояния модуля, а также дополнительное управление скоростью обмена осуществляются с помощью регистра состояния SPSR.

SPSR	7	6	5	4	3	2	1	0
	SPIF	WCOL	-	-	-	-	-	SPI2X
Чтение(R)/Запись(W)	R	R	R	R	R	R	R	R/W
Начальное значение	0	0	0	0	0	0	0	0

Таблица 29. Биты регистра SPSR.

Бит	Название	Описание
7	SPIF	**Флаг прерывания от SPI.** Данный флаг устанавливается в 1 по окончании передачи очередного байта. Если флаг SPIE регистра SPCR установлен в 1 и прерывания разрешены, то одновременно с установкой флага генерируется прерывание от SPI. Также флаг SPIF устанавливается в 1 при переводе микроконтроллера из режима Master в режим Slave посредством вывода $\overline{SS}$ (см. раздел 10.4). Флаг сбрасывается аппаратно либо при старте подпрограммы обработки прерывания, либо после чтения регистра состояния SPI с последующим обращением к регистру данных SPI (SPDR)
6	WCOL	**Флаг конфликта записи.** Данный флаг устанавливается в 1 при попытке записи в регистр данных (SPDR) во время передачи очередного байта. Флаг сбрасывается аппаратно после чтения регистра состояния SPI с последующим обращением к регистру данных SPI
5...1	—	Зарезервированы, читаются как 0
0	SPI2X	**Удвоение скорости обмена.** При установке этого бита в 1 и работе микроконтроллера в режиме Master частота сигнала SCK удваивается

Соединение двух микроконтроллеров (ведущий - ведомый) по интерфейсу SPI показано на рис. 4. Вывод SCK ведущего микроконтроллера является выходом тактового сигнала, а ведомого микроконтроллера — входом.

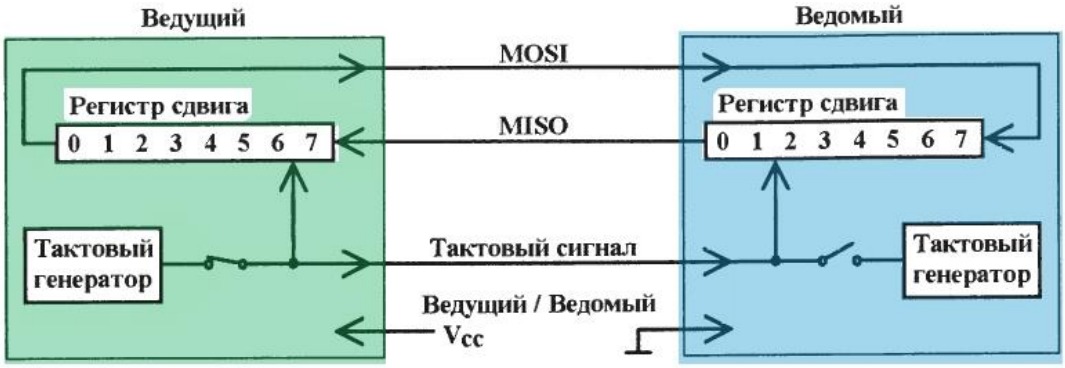

Рисунок 4. Соединение микроконтроллеров по интерфейсу SPI.

Передача данных осуществляется следующим образом. При записи в регистр данных SPI ведущего микроконтроллера запускается генератор тактового сигнала модуля SPI, и данные начинают побитно выдаваться на вывод MOSI и соответственно поступать на вывод MOSI ведомого микроконтроллера. Порядок передачи битов данных определяется состоянием бита DORD регистра SPCR. Если бит установлен в 1, первым передается младший бит байта, если же сброшен в 0 — старший бит. После выдачи последнего бита текущего байта генератор тактового сигнала останавливается с одновременной установкой в 1 флага «Конец передачи» (SPIF). Если прерывания от модуля SPI

разрешены (флаг SPIE регистра SPCR установлен в 1), генерируется запрос на прерывание. После этого ведущий микроконтроллер может начать передачу следующего байта либо, подав на вход SS ведомого напряжение ВЫСОКОГО уровня, перевести его в состояние ожидания.

Одновременно с передачей данных от ведущего к ведомому происходит передача и в обратном направлении, при условии, что на входе SS ведомого присутствует напряжение НИЗКОГО уровня. Таким образом, в каждом цикле сдвига происходит обмен данными между устройствами. Соответственно, в конце каждого цикла флаг SPIF устанавливается в 1 как в ведущем микроконтроллере, так и в ведомом. Принятые байты сохраняются в приемных буферах для дальнейшего использования.

Готовый для передачи байт данных не может быть записан в регистр данных SPI до окончания предыдущего цикла обмена. При попытке изменить содержимое регистра данных во время передачи флаг WCOL регистра SPSR устанавливается в 1. Сбрасывается этот флаг после чтения регистра SPSR с последующим обращением к регистру данных SPI. Соответственно, при приеме данных принятый байт должен быть прочитан из регистра данных SPI до того, как в сдвиговый регистр поступит последний бит следующего байта. В противном случае первый байт будет потерян.

Частота тактового сигнала SCK и соответственно скорость передачи данных по интерфейсу определяются состоянием битов SPR1:SPR0 регистра SPCR и бита SPI2X регистра SPSR (таблица 26). Разумеется, речь идет о микроконтроллере, работающем в режиме Master, так как именно он является источником тактового сигнала. Для устройства, находящегося в режиме Slave, состояние этих битов безразлично.

Таблица 30. Задание частоты тактового сигнала SCK.

SPI2X	SPR1	SPR0	Частота сигнала SCK
0	0	0	$f_{CLK}/4$
0	0	1	$f_{CLK}/16$
0	1	0	$f_{CLK}/64$
0	1	1	$f_{CLK}/128$
1	0	0	$f_{CLK}/2$
1	0	1	$f_{CLK}/8$
1	1	0	$f_{CLK}/32$
1	1	1	$f_{CLK}/64$

Следует иметь в виду, что работа микроконтроллера в режиме Slave гарантируется только на частотах, меньших или равных $f_{CLK}/4$.

Приложение 1. Директивы ассемблера

Компилятор поддерживает ряд директив. Директивы не транслируются непосредственно в код. Вместо этого они используются для указания положения в программной памяти, определения макросов, инициализации памяти и т.д. Список директив приведён в следующей таблице.

Директива Описание
.BYTE ;Зарезервировать байты в ОЗУ
.CSEG ;Программный сегмент
.DB ;Определить байты во флэш или EEPROM
.DEF ;Назначить регистру символическое имя
.DEVICE ;Определить устройство для которого компилируется программа
.DSEG ;Сегмент данных
.DW ;Определить слова во флэш или EEPROM
.ENDM.ENDMACRO ;Конец макроса .EQU ;Установить постоянное выражение
.ESEG ;Сегмент EEPROM
.EXIT ;Выйти из файла
.INCLUDE ;Вложить другой файл
.LIST ;Включить генерацию листинга

.LISTMAC ;Включить разворачивание макросов в листинге
.MACRO ;Начало макроса
.NOLIST ;Выключить генерацию листинга
.ORG ;Установить положение в сегменте
.SET ;Установить переменный символический эквивалент выражения

!!!Все директивы предваряются точкой.

.BYTE - Зарезервировать байты в ОЗУ

Директива BYTE резервирует байты в ОЗУ. Если Вы хотите иметь возможность ссылаться на выделенную область памяти, то директива BYTE должна быть предварена меткой. Директива принимает один обязательный параметр, который указывает количество выделяемых байт. Эта директива может использоваться только в сегменте данных (смотреть директивы CSEG и DSEG). Выделенные байты не инициализируются.

```
Синтаксис:
МЕТКА: .BYTE выражение
Пример:
.DSEG
var1: .BYTE 1 ; резервирует 1 байт для var1
table: .BYTE tab_size ; резервирует tab_size байт
.CSEG
ldi r30,low(table) ; Загружает младший байт регистра Z
ldi r31,high(table) ; Загружает старший байт регистра Z
ld r1,Z ; Загружает VAR1 в регистр 1
```

.CSEG - Программный сегмент

Директива CSEG определяет начало программного сегмента. Исходный файл может состоять из нескольких программных сегментов, которые объединяются в один программный сегмент при компиляции. Программный сегмент является сегментом по умолчанию. Программные сегменты имеют свои собственные счётчики положения которые считают не побайтно, а по словно. Директива ORG может быть использована для размещения кода и констант в необходимом месте сегмента. Директива CSEG не имеет параметров.

```
    Синтаксис:  .CSEG
    Пример:
.DSEG ; Начало сегмента данных
vartab: .BYTE 4 ; Резервирует 4 байта в ОЗУ
.CSEG ; Начало кодового сегмента
const: .DW 2 ; Разместить константу 0x0002 в памяти программ
mov r1,r0 ; Выполнить действия
```

.DB - Определить байты во флэш или EEPROM

Директива DB резервирует необходимое количество байт в памяти программ или в EEPROM. Если Вы хотите иметь возможность ссылаться на выделенную область памяти, то директива DB должна быть предварена меткой. Директива DB должна иметь хотя бы один параметр. Данная директива может быть размещена только в сегменте программ (CSEG) или в сегменте EEPROM (ESEG). Параметры передаваемые директиве - это последовательность

выражений разделённых запятыми. Каждое выражение должно быть или числом в диапазоне (-128..255), или в результате вычисления должно давать результат в этом же диапазоне, в противном случае число усекается до байта, причём БЕЗ выдачи предупреждений. Если директива получает более одного параметра и текущим является программный сегмент, то параметры упаковываются в слова (первый параметр - младший байт), и если число параметров нечётно, то последнее выражение будет усечено до байта и записано как слово со старшим байтом равным нулю, даже если далее идет ещё одна директива DB.

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

.DEF - Назначить регистру символическое имя

Директива DEF позволяет ссылаться на регистр через некоторое символическое имя. Назначенное имя может использоваться во всей нижеследующей части программы для обращений к данному регистру. Регистр может иметь несколько различных имен. Символическое имя может быть переназначено позднее в программе.

```
    Синтаксис:  .DEF Символическое_имя = Регистр
    Пример:
.DEF temp=R16
.DEF ior=R0
.CSEG
ldi temp,0xf0 ; Загрузить 0xf0 в регистр temp (R16)
in ior,0x3f   ; Прочитать SREG в регистр ior (R0)
```

.DEVICE -Определить устройство для которого компилируется программа

Директива DEVICE позволяет указать для какого устройства компилируется программа. При использовании данной директивы компилятор выдаст предупреждение, если будет найдена инструкция, которую не поддерживает данный микроконтроллер. Также будет выдано предупреждение, если программный сегмент, либо сегмент EEPROM превысят размер допускаемый устройством. Если же директива не используется то все инструкции считаются допустимыми, и отсутствуют ограничения на размер сегментов.

```
    Синтаксис: .DEVICE AT90S1200 |AT90S2313 | AT90S2323 | AT90S2333
| AT90S2343 | AT90S4414 | AT90S4433
    Пример:
.DEVICE AT90S1200 ; Используется AT90S1200
.CSEG
push r30 ; Эта инструкция вызовет предупреждение поскольку
           AT90S1200 её не имеет
```

.DSEG - Сегмент данных

Директива DSEG определяет начало сегмента данных. Исходный файл может состоять из нескольких сегментов данных, которые объединяются в один сегмент при компиляции. Сегмент данных обычно состоит только из директив BYTE и меток. Сегменты данных имеют свои собственные побайтные счётчики положения. Директива ORG может быть использована для размещения переменных в необходимом месте ОЗУ. Директива не имеет параметров.

.DW - Определить слова во флэш или EEPROM

Директива DW резервирует необходимое количество слов в памяти программ или в EEPROM. Если Вы хотите иметь возможность ссылаться на выделенную область памяти, то директива DW должна быть предварена меткой. Директива DW должна иметь хотя бы один параметр. Данная директива может быть размещена только в сегменте программ (CSEG) или в сегменте EEPROM (ESEG). Параметры передаваемые директиве - это последовательность выражений разделённых запятыми. Каждое выражение должно быть или числом в диапазоне (-32768..65535), или в результате вычисления должно давать результат в этом же диапазоне, в противном случае число усекается до слова, причем БЕЗ выдачи предупреждений.

Синтаксис:

```
МЕТКА: .DW expressionlist
```

Пример:

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

.ENDMACRO - Конец макроса

Директива определяет конец макроопределения, и не принимает никаких параметров. Для информации по определению макросов смотрите директиву MACRO.

Синтаксис:

```
.ENDMACRO
```

Пример:

```
.MACRO SUBI16 ; Начало определения макроса
subi r16,low(@0) ; Вычесть младший байт первого параметра
sbci r17,high(@0) ; Вычесть старший байт первого параметра
.ENDMACRO
```

.EQU – Присвоить символьное имя

Директива EQU присваивает метке значение. Эта метка может позднее использоваться в выражениях. Метка которой присвоено значение данной директивой не может быть переназначена и её значение не может быть изменено.

```
Синтаксис: .EQU метка = выражение
```

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

.ESEG - Сегмент EEPROM

Директива ESEG определяет начало сегмента EEPROM. Исходный файл может состоять из нескольких сегментов EEPROM, которые объединяются в один сегмент при компиляции. Сегмент EEPROM обычно состоит только из директив DB, DW и меток. Сегменты EEPROM имеют свои собственные побайтные счётчики положения. Директива ORG может быть использована для размещения переменных в необходимом месте EEPROM. Директива не имеет параметров.

```
Синтаксис: .ESEG
Пример:
.DSEG ; Начало сегмента данных
    table: .BYTE tab_size ; зарезервировать tab_size байт.
.ESEG
eevar1: .DW 0xffff ; проинициализировать 1 слово в EEPROM
```

.EXIT – Завершить ассемблирование

Встретив директиву .EXIT, компилятор прекращает компиляцию данного файла. Если директива использована во вложенном файле (см. директиву INCLUDE), то компиляция продолжается со строки следующей после директивы INCLUDE. Если же файл не является вложенным, то компиляция прекращается.

```
Синтаксис:  .EXIT
```

.INCLUDE - Вложить другой файл

Встретив директиву INCLUDE компилятор открывает указанный в ней файл, компилирует его пока файл не закончится или не встретится директива EXIT, после этого продолжает компиляцию начального файла со строки следующей за директивой INCLUDE. Вложенный файл может также содержать директивы INCLUDE.

```
Синтаксис: .INCLUDE "имя_файла"
Пример:  файл iodefs.asm:
.EQU sphigh = 0x3e ; Старший байт указателя стека
.EQU splow = 0x3d ; Младший байт указателя стека файл incdemo.asm
.INCLUDE iodefs.asm ; Вложить определения портов
in r0,sreg ; Прочитать регистр статуса
```

.LIST - Включить генерацию листинга

Директива LIST указывает компилятору на необходимость создания листинга. Листинг представляет из себя комбинацию ассемблерного кода, адресов и кодов операций. По умолчанию генерация листинга включена,

однако данная директива используется совместно с директивой NOLIST для получения листингов отдельных частей исходных файлов.

Синтаксис: .LIST

Пример:

```
.NOLIST ; Отключить генерацию листинга
.INCLUDE "macro.inc" ; Вложенные файлы не будут
.INCLUDE "const.def" ; отображены в листинге
.LIST ; Включить генерацию листинга
```

.LISTMAC - Включить разворачивание макросов в листинге

После директивы LISTMAC компилятор будет показывать в листинге содержимое макроса. По умолчанию в листинге показывается только вызов макроса и передаваемые параметры.

Синтаксис: .LISTMAC

Пример:

```
.MACRO MACX ; Определение макроса
    add r0,@0 ; Тело макроса
    eor r1,@1
    .ENDMACRO ; Конец макроопределения
    .LISTMAC ; Включить разворачивание макросов
    MACX r2,r1 ; Вызов макроса (в листинге будет показано теломакроса)
```

.MACRO - Начало макроса

С директивы MACRO начинается определение макроса. В качестве параметра директиве передаётся имя макроса. При встрече имени макроса позднее в тексте программы, компилятор заменяет это имя на тело макроса. Макрос может иметь до 10 параметров, к которым в его теле обращаются через @0 - @9. При вызове параметры перечисляются через запятые. Определение макроса заканчивается директивой ENDMACRO. По умолчанию в листинг включается только вызов макроса, для разворачивания макроса необходимо использовать директиву LISTMAC. Макрос в листинге показывается знаком +.

Синтаксис: .MACRO макроимя

Пример:

```
.MACRO SUBI16 ; Начало макроопределения
subi @1,low(@0) ;Вычесть младший байт параметра 0 из параметра 1
sbci @2,high(@0) ; Вычесть старший байт параметра 0 из параметра 2
.ENDMACRO ; Конец макроопределения
.CSEG ; Начало программного сегмента
SUBI16 0x1234,r16,r17 ; Вычесть 0x1234 из r17:r16
```

.ORG - Установить положение в сегменте

Директива ORG устанавливает счётчик положения равным заданной величине, которая передаётся как параметр. Для сегмента данных она устанавливает счётчик положения в SRAM (ОЗУ), для сегмента программ это программный счётчик, а для сегмента EEPROM это положение в EEPROM. Если директиве предшествует метка (в той же строке) то метка размещается по адресу указанному в параметре директивы. Перед началом компиляции программный счётчик и счётчик EEPROM равны нулю, а счётчик ОЗУ равен 32

(поскольку адреса 0-31 заняты регистрами). Обратите внимание что для ОЗУ и EEPROM используются побайтные счётчики а для программного сегмента - пословный.

```
    Синтаксис: .ORG выражение
    Пример:
              .DSEG ; Начало сегмента данных
              .ORG 0x37 ; Установить адрес SRAM равным 0x37
    variable: .BYTE 1 ; Зарезервировать байт по адресу 0x37H
              .CSEG
.ORG 0x10 ; Установить программный счётчик равным 0x10
mov r0,r1 ; Данная команда будет размещена по адресу 0x10
```

.SET - Установить переменный символический эквивалент выражения

Директива SET присваивает имени некоторое значение. Это имя позднее может быть использовано в выражениях. Причем в отличии от директивы EQU значение имени может быть изменено другой директивой SET.

```
    Синтаксис: .SET имя = выражение
    Пример:
    .SET io_offset = 0x23
    .SET porta = io_offset + 2
    .CSEG ; Начало кодового сегмента
    clr r2 ; Очистить регистр 2
.SET porta = 0x20
In r2, porta
```

Приложение 2. Система команд. Сводные таблицы команд

Таблица 31. группа команд логических операций

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

Таблица 32. группа команд арифметических операций

Мнемоника	Описание	Операция	Флаги

ADD Rd.Rr	Сложение двух РОН	Rd = Rd + Rr	Z,C,N,V,H
ADC Rd.Rr	Сложение двух РОН с переносом	Rd = Rd + Rr + C	Z,C,N,V,H
ADIW Rdh : Rdl, K	Сложение регистровой пары с константой	Rdh:Rdl =Rdh:Rdl+K	Z,C,N,V,S
SUB Rd,Rr	Вычитание двух РОН	Rd = Rd - Rr	Z,C,N,V,H
SUBI Rd.K	Вычитание константы из РОН	Rd = Rd - K	Z,C,N,V,H
SBC Rd.Rr	Вычитание двух РОН с заемом	Rd = Rd-Rr-C	Z,C,N,V,H
SBCI Rd.K	Вычитание константы из РОН с заемом	Rd = Rd-K-C	Z,C,N,V,H
SBIW Rdh:Rdl,K	Вычитание константы из регистровой пары	Rdh:Rdl=Rdh:Rdl-K	Z,C,N,V,S
DEC Rd	Декрементирование РОН	Rd = Rd-l	Z.N.V
INC Rd	Инкрементирование РОН	Rd = Rd + l	Z,N,V
ASR Rd	Арифметический сдвиг вправо	Rd(n) = Rd(n+l), и = 0..6	Z,C,N,V
LSL Rd	Логический сдвиг влево	Rd(w+l) = Rd(w), Rd(0) = 0	Z,C,N,V
LSR Rd	Логический сдвиг вправо	Rd(n) = Rd(n+l), Rd(7) = 0	Z,C,N,V
ROL Rd	Сдвиг влево через перенос	Rd(0) = C, Rd(w+l) = Rd(w), C = Rd(7)	Z,C,N,V
ROR Rd	Сдвиг вправо через перенос	Rd(7) = C, Rd(w) = Rd(w+l), C = Rd(0)	Z,C,N,V
MUL Rd,Rr	Умножение беззнаковых чисел	Rl:RO = RdxRr	Z,C
MULS Rd,Rr	Умножение чисел со знаком	Rl:RO = RdxRr	Z,C
MULSU Rd.Rr	Умножение беззнакового числа на число со знаком	Rl:RO = RdxRr	Z,C
FMUL Rd.Rr	Умножение дробных беззнаковых чисел	Rl:RO = (RdxRr)«l	Z,C
FMULS Rd,Rr	Умножение дробных чисел со знаком	Rl:RO = (RdxRr)«l	Z,C
FMULSU Rd.Rr	Умножение дробного беззнакового числа и дробного числа со знаком	Rl:RO = (RdxRr)«l	Z,C

Таблица 33. группа команд операций с битами

Мнемоника	**Описание**	**Операция**	**Флаги**
CBR Rd,K	Сброс бита(ов)РОН	Rd=R<U($FF-K)	Z.N.V
SBR Rd, K	Установка бита(ов) РОН	Rd = RdvK	Z,N,V
CBI A, b	Сброс бита РВВ	A.b = 0	-
SBI A, b	Установка бита РВВ	A.b=l	-
BCLR s	Сброс флага	SREG.s = 0	SREG, s
BSET s	Установка флага	SREG.s=l	SREG, s
BLD Rd, b	Загрузка бита РОН из флага T (SREG)	Rd.b = T	-
BST Rr, b	Запись бита РОН в флаг T (SREG)	T = Rr.b	T
CLC	Сброс флага переноса	C = 0	C
SEC	Установка флага переноса	C = l	C
CLN	Сброс флага отрицательного числа	N = 0	N
SEN	Установка флага отрицательного числа	N=1	N
CLZ	Сброс флага нуля	Z = 0	Z
SEZ	Установка флага нуля	Z=l	Z
CLI	Обшее запрещение прерываний	1 = 0	I
SEI	Общее разрешение прерываний	1=1	I
CLS	Сброс флага знака	**s=o**	S

SES	Установка флага знака	S = l	S
CLV	Сброс флага переполнения дополнительного кода	**v=o**	V
SEV	Установка флага переполнения дополнительного кода	V=l	V
CLT	Сброс флага T	T = 0	T
SET	Установка флага T	T=l	T
CLH	Сброс флага половинного переноса	H = 0	H
SEH	Установка флага половинного переноса	H=l	H

Таблица 34. группа команд пересылки данных

Мнемоника	**Описание**	**Операция**	**Флаги**
MOV Rd, Rr	Пересылка между РОН	Rd = Rr	-
MOVW Rd, Rr	Пересылка 2-байтных значений	Rd+l:Rd = Rr+l:Rr	-
LDI Rd, K	Загрузка константы в РОН	Rd = K	-
LD Rd, X	Косвенное чтение	Rd = [X]	-
LD Rd, X+	Косвенное чтение с постинкрементом	Rd = [X],X=X+l	-
LD Rd, -X	Косвенное чтение с преддекрементом	X = X-t,Rd = [X]	-
LD Rd, Y	Косвенное чтение	Rd = [Y]	-
LD Rd, Y+	Косвенное чтение с постинкрементом	Rd = [Y],Y = Y+l	-
LD Rd,-Y	Косвенное чтение с преддекрементом	Y = Y-l,Rd = [Y]	-
LDD Rd, Y+ q	Косвенное относительное чтение	Rd = [Y+q]	-
LD Rd, Z	Косвенное чтение	Rd = [Z]	-
LD Rd, Z+	Косвенное чтение с постинкрементом	Rd = [Z],Z = Z + l	-
LD Rd, -Z	Косвенное чтение с преддекрементом	Z = Z-l,Rd = [Z]	-
LDD Rd, Z+q	Косвенное относительное чтение	Rd = [Z+q]	-
LDS Rd, k	Непосредственное чтение из ОЗУ	Rd = [kJ	-
ST X, Rr	Косвенная запись	[X] = Rr	-
ST X+, Rr	Косвенная запись с постинкрементом	[X] = Rr,X=X+l	-
ST -X, Rr	Косвенная запись с преддекрементом	X = X-l,[X] = Rr	-
ST Y, Rr	Косвенная запись	[Y] = Rr	-
ST Y+, Rr	Косвенная запись с постинкрементом	[Y] = Rr,Y=Y+l	-
ST -Y, Rr	Косвенная запись с преддекрементом	Y=Y-l,[XJ = Rr	-
STD Y+q, Rr	Косвенная относительная запись	[Y+q] = Rr	-
ST Z, Rr	Косвенная запись	[Z] = Rr	-
ST Z+, Rr	Косвенная запись с постинкрементом	[Z] = Rr,Z = Z + l	-
ST -Z, Rr	Косвенная запись с преддекрементом	Z = Z-l,[Z]=Rr	-
STD Z+ q, Rr	Косвенная относительная запись	[Z+q] = Rr	-
STS k, Rr	Непосредственная запись в ОЗУ	[k] = Rr	-
LPM	Загрузка данных из памяти программ	R0 = {Z}	-
LPM Rd, Z	Загрузка данных из памяти программ	RB = {Z}	-
LPM Rd, Z+	Загрузка данных из памяти программ с постинкрементом	RB = {Z},Z = Z+I	-
ELPM	Расширенная загрузка данных из памяти программ	R0 = {RAMPZ:Z}	-
ELPM Rd, Z	Расширенная загрузка данных из памяти программ	RB = {RAMPZ:Z}	-
ELPM Rd, Z+	Расширенная загрузка данных из памяти программ с постинкрементом	RB = {RAMPZ:Z}, RAMPZ:Z=	-

SPM	Запись в память программ	{Z} = R1:R0	-
IN Rd, A	Пересылка из РВВ в РОН	RD = A	-
0UT A,Rr	Пересылка из РОН в РВВ	A=RR	-
PUSH Rr	Сохранение байта в стеке	STACK = RR	-
POP~Rd	Извлечение байта из стека	RD = STACK	—

Таблица 35. группа команд передачи управления

Мнемоника	**Описание**	**Операция**	Флаги
RJMP k	Относительный безусловный переход	PC = PC + к+1	-
IJMP	Косвенный безусловный переход	PC = Z	-
EIJMP	Расширенный косвенный безусловный переход	PC = EIND:Z	-
JMP k	Абсолютный переход	PC = к	-
RCALL k	Относительный вызов подпрограммы	PC = PC + к + 1	-
ICALL	Косвенный вызов подпрограммы	PC = Z	-
EICALL	Расширенный косвенный вызов подпрограммы	PC = EIND:Z	-
CALL k	Абсолютный вызов подпрограммы	PC = к	-
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
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 + к+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 + к+1	-
BRGE	Переход по «больше или равно» (числа со знаком)	Если (N⊕V) = 0, то PC = PC + к+1	-
BRLT	Переход по «меньше нуля» (числа со знаком)	Если (N⊕V) = 1, то PC = PC + к+1	-
BRHS	Переход по половинному переносу	Если H= 1, то PC = PC + к+1	-
BRHC	Переход, если нет половинного переноса	Если H = 0, то PC = PC + к+1	-
BRTS	Переход, если флаг T установлен	ЕслиT=1, то PC = PC + к+1	-
BRTC	Переход, если флаг T сброшен	Если T = 0, то PC = PC + к+1	-
BRVS	Переход по переполнению дополнительного кода	Если V=1, то PC = PC + к+1	-
BRVC	Переход, если нет переполнения дополнительного кода	Если V = 0, то PC = PC + к + 1	-
BRID	Переход, если прерывания запрещены	Если I = 0, то PC = PC + к+1	-
BRIE	Переход, если прерывания разрешены	Если I = 1, то PC = PC + к+1	-

Таблица 36. группа команд управления системой

Мнемоника	**Описание**	**Операция**	**Флаги**
NOP	Нет операции		-
SLEEP	Переход в «спящий» режим	См. соответствующие разделы	-
WDR	Сброс сторожевого таймера	См. соответствующие разделы	-
BREAK	Останов	Используется только внутри-схемным отладчиком	-

Обозначение, символ	**Описание**
Регистр состояния	
SREG	Регистр состояния микроконтроллера
C	Флаг переноса (0-й бит регистра SREG)
Z	Флаг нуля (1-й бит регистра SREG)
N	Флаг отрицательного значения (2-й бит регистра SREG)
V	Флаг переполнения дополнительного кода

	(3-й бит регистра SREG)
S	Флаг знака (4-й бит регистра SREG); S = N ffi V
H	Флаг половинного переноса (5-й бит регистра SREG)
T	Пользовательский флаг (6-й бит регистра SREG)
I	Флаг общего разрешения прерываний (7-й бит регистра SREG)
Регистры и операнды	
Rd	Регистр-приемник (иногда также регистр-источник) в регистровом файле
Rr	Регистр-источник в регистровом файле
K	Константа (данные)
к	Адрес — константа
b	Номер бита РОН или РВВ (0...7)
s	Номер бита регистра состояния SREG (0...7)
X,Y,Z	Индексные регистры (X = R27:R26, Y = R29.R28, Z = R31 :R30)
I/O	Регистр ввода/вывода
A	Адрес в пространстве ввода/вывода
q	Смещение при относительной косвенной адресации (6-битное значение)
Операции	
•	Логическое И
V	Логическое ИЛИ
®	Исключающее ИЛИ
Система	
PC	Счетчик команд
STACK	Текущий уровень стека
SP	Указатель стека

Содержание

Список использованных источников

1. Голубцов М. С. Микроконтроллеры AVR: от простого к сложному. – М.: СОЛОН-Пресс, 2003. 288 с
2. Евстифеев А.В. Микроконтроллеры AVR семейства Mega. Руководство пользователя. — М.: Издательский дом «Додэка-XXI», 2007. — 592 с: ил.
3. Ревич Ю.В. Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера.- Спб.: БХВ-Петербург,2008. 384 с
4. Хартов В.Я. Микроконтроллеры AVR. Практикум для начинающих. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2007. - 240 с: ил.