МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

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

ТЕОРИЯ И ПРИНЦИПЫ ПОСТРОЕНИЯ ЦИФРОВЫХ СИСТЕМ УПРАВЛЕНИЯ

Учебно-методическое пособие

Казань
2025

Теория и принципы построения цифровых систем управления : учебно-
М74 методическое пособие / составитель М. Ф. Садыков. – Казань : КГЭУ, 2025. – 125 с.

Содержит требования к структуре и содержанию курсового проекта по дисциплине «Теория и принципы построения цифровых систем управления», выполнение которого способствует развитию критического мышления, формированию исследовательских навыков и готовности решать прикладные задачи профессиональной направленности.

Предназначено для обучающихся по образовательной программе направления подготовки 13.03.02 Электроэнергетика и электротехника.

УДК
ББК

Рекомендовано к изданию Учебно-методическим советом
Института электротехники и электроники

Оглавление

Введение……..……. 4

Глава 1. Знакомство с семейством Mega…….……..…. 6
1.1. Общие сведения…….……... 6
1.2. Архитектура микроконтроллеров семейства Mega. Общие сведения……..…….. 13
1.3. Способы адресации памяти данных…….……...… 36
1.4. Прерывания.Общие сведения…… 41
Вопросы по 1 главе …… 46

Глава 2. Порты ввода/вывода ……..…..... 47
2.1. Общие сведения……... 47
2.2. Таймеры…….……... 55
2.3. Прерывания от таймеров/счетчиков…….. 58
2.4. Предделители таймеров/счетчиков ……...…… 63
2.5. Восьмибитные таймеры/ счетчики…….…… 67
2.6. Вопросы по 2 главе …… 74

Глава 3. Микроконтроллеры ……...……..... 75
3.1.Управление тактовым сигналом …….. 75
3.2.Асинхронный режим……..…… 82
3.3.Обращение к 16-битным регистрам ……..…. 93
Вопросы по 3 главе …… 105

Глава 4. Сторожевой таймер. Модулятор…… 106
4.1.Общие сведения……...…… 106
4.2.Сторожевой таймер……. 107
Вопросы по 4 главе ……... 116
Список литературы……..……... 117

Приложение 1……..…….……. 118
Приложение 2……..…….……. 121
Приложение 3……..…….……. 122

Введение

Микроконтроллеры и микропроцессорные системы в настоящее время занимают лидирующие позиции в области решения задач автоматического управления различными процессами и объектами. Возможность реализации на их основе локального «электронного мозга» открывает практически неисчерпаемые перспективы по созданию интеллектуальных устройств. Диапазон использования микропроцессорных систем в современной технике безграничен: от бытовых приборов до сложных космических систем. Высокая надежность и постоянно снижающаяся стоимость микроконтроллеров и микропроцессоров являются решающими факторами в пользу их выбора для элементной базы систем управления самого разнообразного назначения.

В настоящее время выпускается широкий набор микроконтроллеров различной разрядности и быстродействия. По области применения, структурной организации, разрядности, набору периферийных устройств, системе команд и прочим признакам микроконтроллеры сгруппированы в семейства, число которых достаточно велико. К наиболее ярким представителям различных семейств относятся микроконтроллеры семейства AVR фирмы Atmel (Microchip), PIC-контроллеры фирмы MicroChip, ARM-контроллеры различных производителей и др.

Предметом дисциплины является применение микроконтроллеров в современных электронных устройствах и системах различного назначения, а также в автоматизированных системах контроля и управления.

Изучение основ микропроцессорной техники рассматривается на примере универсального 8-ми разрядного RISC-контроллера ATmega48x/88x/168x семейства AVR.

Курсовой проект по дисциплине «Теория и принципы построения цифровых систем управления» предполагает проектирование электронных устройств и систем на основе программируемых микроконтроллеров универсального назначения, включает разработку схемотехнических решений, а также разработку и отладку его программного обеспечения. Инженер электронной техники должен знать основы построения и функционирования встраиваемых систем, связанных с контролем, управлением и автоматизацией технологических процессов на базе современных микроконтроллеров.

Разрабатываемое микроконтроллерное устройство (МКУ) представляет собой цифровую систему управления, включает в себя совокупность аппаратных (Hard Ware) и программных средств (Soft Ware). Аппаратные средства обеспечивают максимальную производительность или быстродействие, а программные средства – расширение круга задач, решаемых данной системой. Как правило, реализация вычислений программными средствами является менее быстродействующей. Поэтому в процессе проектирования микропроцессорных устройств, в целом необходимо

определить оптимальные соотношения и распределение функций между аппаратными и программными средствами с целью получения заданных характеристик системы. При этом в процессе проектирования производительность системы, объем памяти, габариты, потребляемая энергия выступают в качестве критериев выбора конкретной структуры разрабатываемого МКУ. Обобщающим критерием выбора структуры МКУ при решении конкретной задачи, как правило, является стоимостный критерий, сочетающий в себе все перечисления выше.

Курсовое проектирование должно включать следующие этапы проектирования:

1. Изучение теоретического материала по возможностям и особенностям заданного микроконтроллера;
2. Обоснование выбора модулей в составе заданного микроконтроллера, необходимых для конкретного применения;
3. Разработка структуры и алгоритма функционирования разрабатываемого МКУ;
4. Разработка, построение и описание функциональной схемы МКУ;
5. Обоснование выбора и схемотехнического решения аппаратной части МКУ;
6. Описание элементной базы и используемых микросхем;
7. Расчет электрического сопряжения элементов МКУ;
8. Разработка и описание структуры программного обеспечения МКУ, его реализация на языке ассемблера для заданного микроконтроллера;
9. Отладки разработанной программы на языке ассемблера в интегрированной среде разработки.

Глава 1. Знакомство с семейством Mega

1.1. Общие сведения

Как и все микроконтроллеры AVR фирмы Atmel, микроконтроллеры семейства Mega являются 8-битными микроконтроллерами, предназначенными для использования во встраиваемых приложениях. Они изготавливаются по малопотребляющей КМОП-технологии, которая в сочетании с усовершенствованной RISC-архитектурой позволяет достичь наилучше- го соотношения стоимость/быстродействие/энергопотребление. Микроконтроллеры описываемого семейства являются наиболее развитыми представителями микроконтроллеров AVR общего применения.

Отличительные особенности

К особенностям микроконтроллеров AVR семейства Mega можно отнести:

- FLASH-память программ объемом от 8 до 256 Кбайт (число циклов стирания/записи не менее 10 000);
- оперативная память (статическое ОЗУ) объемом от 512 байт до 8 Кбайт;
- память данных на основе ЭСППЗУ (EEPROM) объемом от 256 байт до 4 Кбайт (число циклов стирания/записи не менее 100 000);
- возможность защиты от чтения и модификации памяти программ и данных;
- возможность программирования непосредственно в системе через последовательные интерфейсы SPI и JTAG;
- возможность самопрограммирования;
- возможность внутрисхемной отладки в соответствии со стандартом IEEE 1149.1 (JTAG), а также наличие собственного однопроводного интерфейса внутрисхемной отладки debugWire1);
- разнообразные способы синхронизации: встроенный RC-генератор с внутренней или внешней времязадающей RC-цепочкой, встроенныйгенератор с внешним кварцевым или пьезокерамическим резонато- ром, внешний сигнал синхронизации;
- наличие нескольких режимов пониженного энергопотребления;
- наличие детектора пониженного напряжения питания (Brown-Out Detector — BOD);
- возможность программного снижения частоты тактового генератора).

Архитектура ядра

Ядро микроконтроллеров AVR семейства Mega выполнено по усовершенствованной RISC-архитектуре (enhanced RISC) (Рис. 1.1), в которой используется ряд решений, направленных на повышение быстродействия

микроконтроллеров.

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

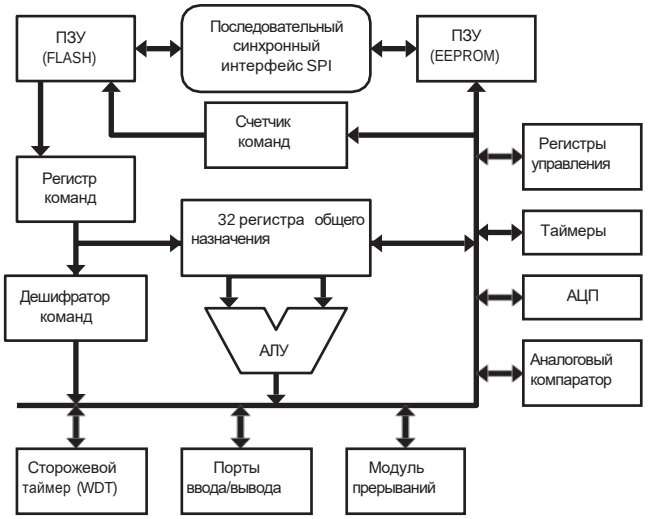


Рис. 1.1. Архитектура ядра микроконтроллеров AVR

В микроконтроллерах AVR реализована Гарвардская архитектура, характеризующаяся раздельной памятью программ и данных, каждая из ко- торых имеет собственные шины доступа. Такая организация позволяет одновременно работать как с памятью программ, так и с памятью данных. Разделение информационных шин позволяет использовать для каждого типа памяти шины различной разрядности, причем способы адресации и доступа к каждому типу памяти также различаются. В сочетании с двух- уровневым конвейером команд такая архитектура позволяет достичь про- изводительности в 1 MIPS на каждый МГц тактовой частоты.

Цоколевка и описание выводов

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

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

- ATmega48, ATmega48V (Рис. 1.3) — имеют FLASH-память программ объемом 4 Кбайт, ОЗУ объемом 512 байт и EEPROM-память данных объемом 256 байт;
- ATmega88, ATmega88V (Рис. 1.3) — имеют FLASH-память программ объемом 8 Кбайт, ОЗУ объемом 1 Кбайт и EEPROM-память данных объемом 512 байт;
- ATmega168, ATmega168V (Рис. 1.3) — имеют FLASH-память программ объемом 16 Кбайт, ОЗУ объемом 1 Кбайт и EEPROM-память

данных объемом 512 байт.

Таблица 1.1. Основные параметры микроконтроллеров AVR семейства Mega

Обозначение	Память программ (FLASH) [Кбайт]	Память данных (ОЗУ) [байт]	Память данных (EEPROM) [байт]	Количество контактов ввода/вывода	Напряжение питания [В]	Тактовая частота [МГц]	Тип корпуса
ATmega 48	4	512	256	23	2.7…5.5 4.5…5.5	0…10 0…20	DIP-28 TQFP-32 MLF-32
ATmega 48V					1.8…5.5 2.7…5.5	0…4 0…10	
ATmega 88	8	1К	512	23	2.7…5.5 4.5…5.5	0…10 0…20	DIP-28 TQFP-32 MLF-32
ATmega 88V					1.8…5.5 2.7…5.5	0…4 0…10	

Обозначение	Память программ (FLASH) [Кбайт]	Память данных (ОЗУ) [байт]	Память данных (EEPROM) [байт]	Количество контактов ввода/вывода	Напряжение питания [В]	Тактовая частота [МГц]	Тип корпуса
ATmega168	16	1К	512	23	2.7…5.5 4.5…5.5	0…10 0…20	DIP-28 TQFP-32 MLF-32

В Табл. 1.2 для каждой линейки микроконтроллеров приведены обозначения выводов и указаны их функции (как основные, так и допол- нительные). Кроме того, для каждого вывода в таблицах указан его тип (вход, выход, вход/выход, вывод питания).

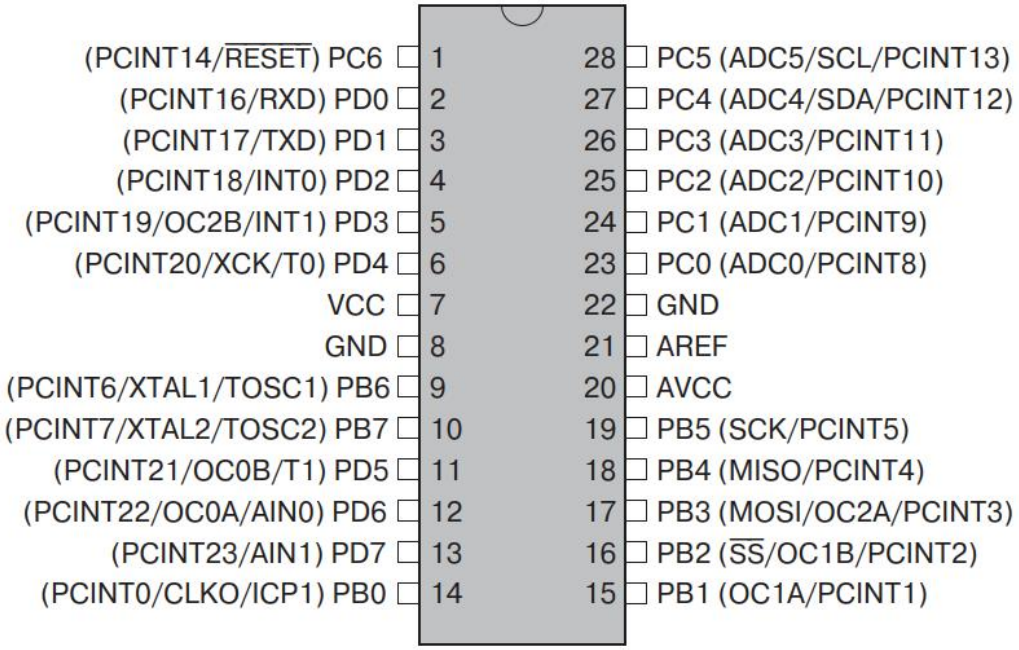
В таблицах использованы следующие обозначения: I — вход;

O — выход;

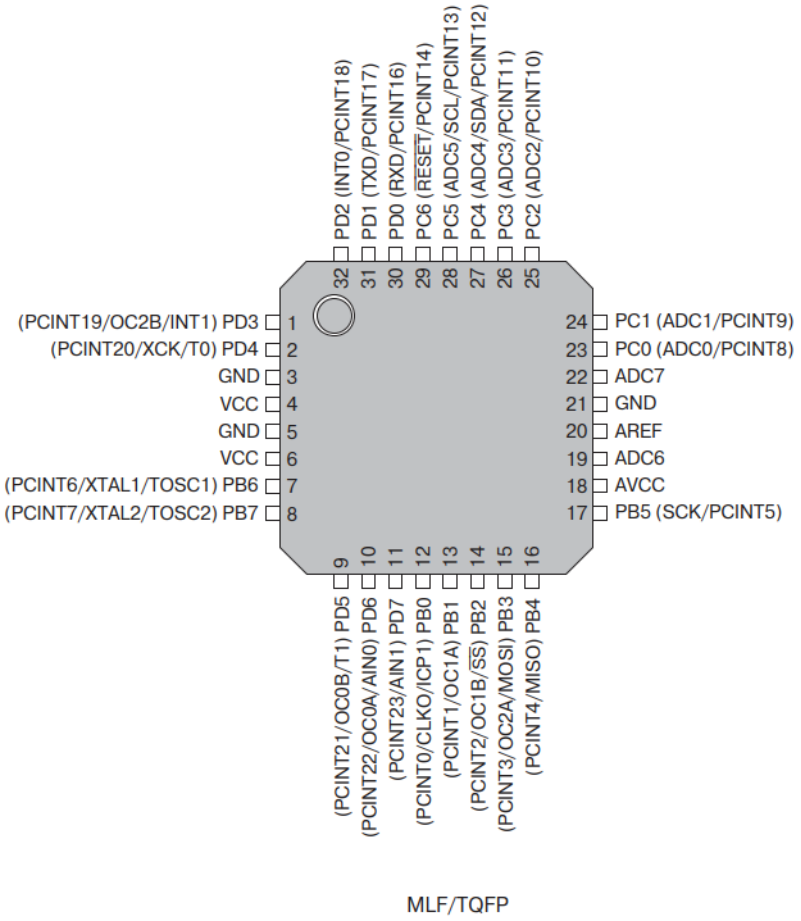
I/O — вход/выход;

P — вывод питания;

ША — шина адреса; ШД — шина данных.



DIP



MLF/TQFP

Рис.1.2.Расположение выводов (вид сверху) моделей ATmega ATmega48x/88x/168x

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP MLF		
Порт B. 8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PB0 (ICP1/CLKO/PCINT0)	14	12	I/O	0-й бит порта B Вход захвата таймера/счетчика T1 Выход предделителя тактового сигнала Вход внешнего прерывания по изменению состояния вывода
PB1 (OC1A/PCINT1)	15	13	I/O	1-й бит порта B Выход A таймера/счетчика T1 Вход внешнего прерывания по изменению состояния вывода
PB2 ($\overline{SS}$ /OC1B/PCINT2)	16	14	I/O	2-й бит порта B Выбор Slave-устройства на шине SPI Выход B таймера/счетчика T1 Вход внешнего прерывания по изменению состояния вывода
PB3 (MOSI/OC2A/PCINT3)	17	15	I/O	3-й бит порта B Выход (Master) или вход (Slave) данных модуля SPI Выход A таймера/счетчика T2 Вход внешнего прерывания по изменению состояния вывода
PB4 (MISO/PCINT4)	18	16	I/O	4-й бит порта B Вход (Master) или выход (Slave) данных модуля SPI Вход внешнего прерывания по изменению состояния вывода
PB5 (SCK/PCINT5)	19	17	I/O	5-й бит порта B Выход (Master) или вход (Slave) тактового сигнала модуля SPI Вход внешнего прерывания по изменению состояния вывода
PB6 (XTAL1/TOSC1/PCINT6)	9	7	I/O	6-й бит порта B Вход тактового генератора Вывод для подключения резонатора к таймеру/счетчику T2 Вход внешнего прерывания по изменению состояния вывода

Табл.1.2. Описание выводов моделей ATmega48x/88x/168x

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP MLF		
PB7 (XTAL2/TOSC2/PCINT7)	10	8	I/O	7-й бит порта B Выход тактового генератора Вывод для подключения резонатора к таймеру/счетчику T2 Вход внешнего прерывания по изменению состояния вывода
Порт C. 7-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PC0 (ADC0/PCINT8)	23	23	I/O	0-й бит порта C Вход АЦП Вход внешнего прерывания по изменению состояния вывода
PC1 (ADC1/PCINT9)	24	24	I/O	1-й бит порта C Вход АЦП Вход внешнего прерывания по изменению состояния вывода
PC2 (ADC2/PCINT10)	25	25	I/O	2-й бит порта C Вход АЦП Вход внешнего прерывания по изменению состояния вывода
PC3 (ADC3/PCINT11)	26	26	I/O	3-й бит порта C Вход АЦП Вход внешнего прерывания по изменению состояния вывода
PC4 (ADC4/SDA/PCINT12)	27	27	I/O	4-й бит порта C Вход АЦП Вход/выход данных модуля TWI Вход внешнего прерывания по изменению состояния вывода
PC5 (ADC5/SCL/PCINT13)	28	28	I/O	5-й бит порта C Вход АЦП Вход/выход тактового сигнала модуля TWI Вход внешнего прерывания по изменению состояния вывода
PC6 ($\overline{\text{RESET}}$ /PCINT14)	1	29	I/O	6-й бит порта C Вход сброса Вход внешнего прерывания по изменению состояния вывода
ADC6	—	19	I	Вход АЦП
ADC7	—	22	I	Вход АЦП

Табл.1.2. Описание выводов моделей ATmega48x/88x/168x (продолжение)

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP MLF		
Порт D. 8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PD0 (RXD/PCINT16)	2	30	I/O	0-й бит порта D Вход USART Вход внешнего прерывания по изменению состояния вывода
PD1 (TXD/PCINT17)	3	31	I/O	1-й бит порта D Выход USART Вход внешнего прерывания по изменению состояния вывода
PD2 (INT0/PCINT18)	4	32	I/O	2-й бит порта D Вход внешнего прерывания Вход внешнего прерывания по изменению состояния вывода
PD3 (INT1/OC2B/PCINT19)	5	1	I/O	3-й бит порта D Вход внешнего прерывания Выход B таймера/счетчика T2 Вход внешнего прерывания по изменению состояния вывода
PD4 (T0/XCK/PCINT20)	6	2	I/O	4-й бит порта D Вход внешнего тактового сигнала таймера/счетчика T0 Вход/выход внешнего тактового сигнала USART Вход внешнего прерывания по изменению состояния вывода
PD5 (T1/OC0B/PCINT21)	11	9	I/O	5-й бит порта D Вход внешнего тактового сигнала таймера/счетчика T1 Выход B таймера/счетчика T0 Вход внешнего прерывания по изменению состояния вывода
PD6 (AIN0/OC0A/PCINT22)	12	10	I/O	6-й бит порта D Неинвертирующий вход компаратора Выход A таймера/счетчика T0 Вход внешнего прерывания по изменению состояния вывода
PD7 (AIN1/PCINT23)	13	11	I/O	7-й бит порта D Инвертирующий вход компаратора Вход внешнего прерывания по изменению состояния вывода

Табл.1.2. Описание выводов моделей ATmega48x/88x/168x(продолжение)

1.2. Архитектура микроконтроллеров семейства Mega. Общие сведения

Микроконтроллеры AVR семейства Mega являются 8-битными микро- контроллерами с RISC-архитектурой. Они имеют в своем составе электри- чески стираемую память программ (FLASH) и данных (EEPROM), а также разнообразные периферийные устройства. Следует отметить, что набор периферийных устройств в микроконтроллерах семейства Mega гораздо богаче, чем в микроконтроллерах семейства Tiny. Более того, состав этих устройств от модели к модели практически не меняется (меняется только количество однотипных модулей и их функциональные возможности). В любой модели имеется хотя бы по одному 8- и 16-битному таймеру/счет- чику, хотя бы по одному интерфейсному модулю USART и SPI, аналого- вый компаратор, сторожевой таймер и, конечно, порты ввода/вывода. К устройствам, присутствующим не во всех моделях семейства, относятся АЦП, модуль двухпроводного интерфейса TWI (Two Wire Interface, аналог шины I^2C), а также модули интерфейсов JTAG и debugWire.

Структурная схема микроконтроллеров ATmega48x/88x/168x при- ведена на **Рис. 1.3**. Особенностями моделей этой линейки являются:

- 3 порта ввода/вывода (порты B, D — 8-битные, порт C — 7-битный);
- вход аппаратного сброса и выводы для подключения резонатора сов- мещены с линиями ввода/вывода;
- два 8-битных (T0, T2) и два 16-битных (T1, T3) таймера/счетчика;
- 6 каналов ШИМ;
- по одному интерфейсному модулю USART, SPI и TWI, причем мо- дуль USART может работать в режиме SPI;
- 6/8-канальный (в зависимости от корпуса) 10-битный АЦП;
- отладочный интерфейс debugWIRE

Организация памяти

В микроконтроллерах AVR семейства Mega реализована Гарвардская архитектура, в соответствии с которой разделены не только адресные про- странства памяти программ и памяти данных, но также и шины доступа к ним. Способы адресации и доступа к этим областям памяти также различ- ны. Такая структура позволяет центральному процессору работать одно- временно как с памятью программ, так и с памятью данных, что сущест- венно увеличивает производительность. Каждая из областей памяти дан- ных (ОЗУ и EEPROM) также расположена в своем адресном пространстве.

Обобщенная карта памяти микроконтроллеров AVR семейства Mega приведена на **Рис. 1.4**.

Обратите внимание на следующее. Поскольку микроконтроллеры AVR имеют 16-битную систему команд, объем памяти программ на рисунке указан не в байтах, а в 16-битных словах. Символ «$» перед числом означа- ет, что это число записано в шестнадцтеричной системе счисления.

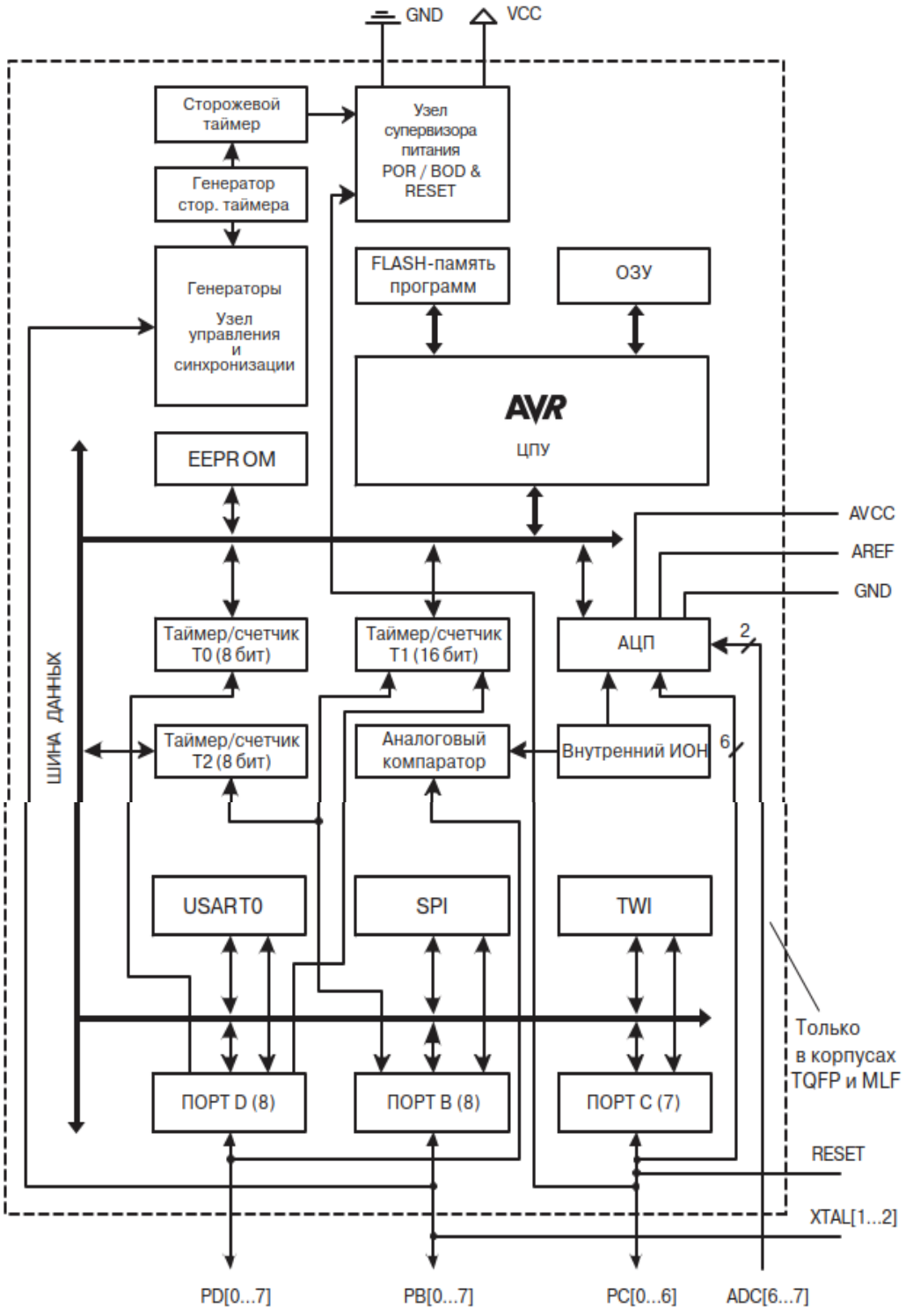
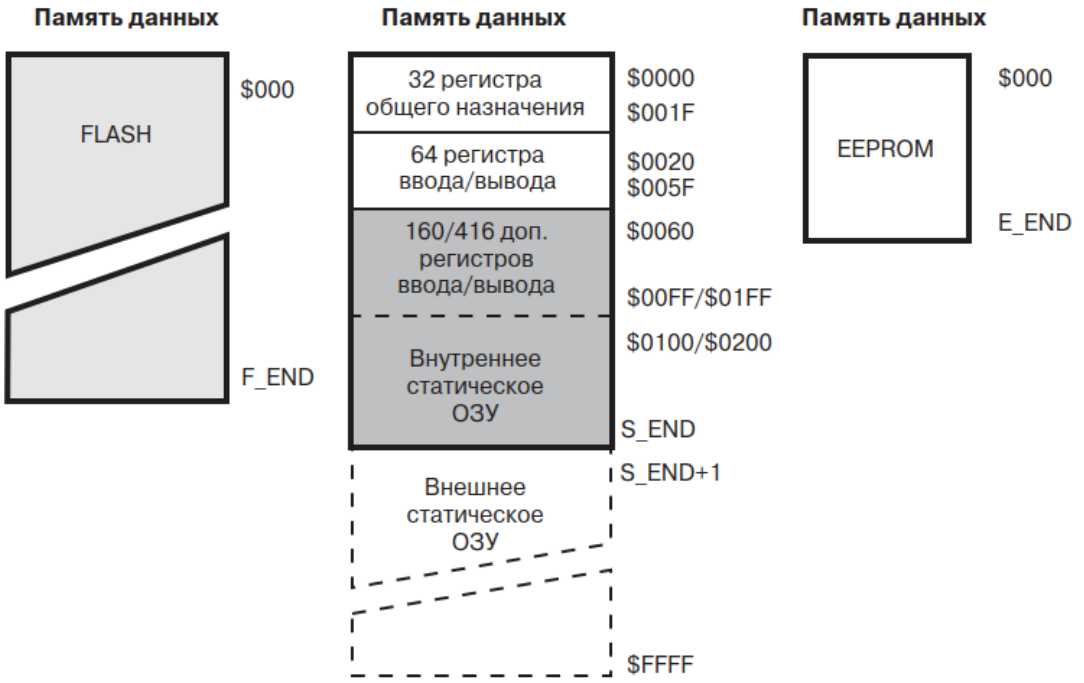


Рис.1.3. . Структурная схема микроконтроллеров ATmega48x/88x/168x



Модель	Память программ (FLASH)		Память данных (ОЗУ)				Память данных (EEPROM)	
	Верхняя граница [F_END]	Объем [слов]	Верхняя граница [S_END]	Объем [байт]	Кол-во доп. регистров ввода/вывода	Внешнее ОЗУ	Верхняя граница [E_END]	Объем [байт]
ATmega48x	$007FF	2 K	$02FF	512	160		$0FF	256
ATmega8515x	$00FFF	4 K	$025F	512	0	•	$1FF	512
ATmega8535x	$00FFF	4 K	$025F	512	0		$1FF	512
ATmega8x	$00FFF	4 K	$045F	1 K	0		$1FF	512
ATmega88x	$00FFF	4 K	$04FF	1 K	160		$1FF	512
ATmega16x	$01FFF	8 K	$045F	1 K	0		$1FF	512
ATmega162x	$01FFF	8 K	$04FF/$045F	1 K	160/0	•	$1FF	512
ATmega164x	$01FFF	8 K	$04FF	1 K	160		$1FF	512
ATmega165x	$01FFF	8 K	$04FF	1 K	160		$1FF	512
ATmega168x	$01FFF	8 K	$04FF	1 K	160		$1FF	512
ATmega32x	$03FFF	16 K	$085F	2 K	0		$3FF	1 K
ATmega324x	$03FFF	16 K	$08FF	2 K	160		$3FF	1 K
ATmega325x	$03FFF	16 K	$08FF	2 K	160		$3FF	1 K
ATmega3250x	$03FFF	16 K	$08FF	2 K	160		$3FF	1 K
ATmega64x	$07FFF	32 K	$10FF	4 K	160	•	$7FF	2 K
ATmega640x	$07FFF	32 K	$21FF	8 K	416	•	$FFF	4 K
ATmega644x	$07FFF	32 K	$10FF	4 K	160		$7FF	2 K
ATmega645x	$07FFF	32 K	$10FF	4 K	160		$7FF	2 K
ATmega6450x	$07FFF	32 K	$10FF	4 K	160		$7FF	2 K
ATmega128x	$0FFFF	64 K	$10FF	4 K	160	•	$FFF	4 K
ATmega1280x	$0FFFF	64 K	$21FF	8 K	416	•	$FFF	4 K
ATmega1281x	$0FFFF	64 K	$21FF	8 K	416	•	$FFF	4 K
ATmega2560x	$1FFFF	128 K	$21FF	8 K	416	•	$FFF	4 K
ATmega2561x	$1FFFF	128 K	$21FF	8 K	416	•	$FFF	4 K

Рис.1.4. Карта памяти микроконтроллеров семейства Mega

Память программ

Память программ предназначена для хранения команд, управляющих работой микроконтроллера. Память программ также часто используется для хранения таблиц констант, не меняющихся во время работы программы.

Как было сказано выше, память программ представляет собой электрически стираемое ППЗУ (FLASH-ПЗУ). В связи с тем, что длина всех команд кратна одному слову (16 бит), память программ имеет 16-битную

организацию. Соответственно, объем памяти микроконтроллеров семейства составляет от 4К (4 1024) до 64К (64 1024) 16-битных слов. В подавляющем большинстве моделей микроконтроллеров семейства Mega память программ логически разделена на две неравные части: область прикладной программы и область загрузчика. В последней может располагаться специальная программа (загрузчик), позволяющая микроконтроллеру самостоятельно управлять загрузкой и выгрузкой прикладных программ.

Для адресации памяти программ используется счетчик команд (Program Counter — PC). Размер счетчика команд составляет от 11 до 17 бит, в зависимости от объема адресуемой памяти.

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

При возникновении прерывания после сохранения в стеке текущего значения счетчика команд происходит выполнение команды, расположенной по адресу соответствующего вектора. Поэтому по данным адресам располагаются команды перехода к подпрограммам обработки прерываний. В моделях с памятью программ небольшого объема (8 Кбайт и менее) в таблицах векторов прерываний используются команды относительного перехода (RJMP), а в остальных моделях — команды абсолютного перехода (JMP).

В большинстве микроконтроллеров семейства Mega положение векто- ра сброса и/или таблицы векторов прерываний может быть изменено. Они могут располагаться не только в начале памяти программ, как описано вы- ше, но и в начале области загрузчика.

Как известно, память программ может использоваться не только для хранения кода программы, но также и для хранения различных констант. Для пересылки байта из памяти программ в память данных существуют две специальных команды — LPM и ELPM (последняя есть только в моделях, имеющих память программ 128 Кбайт и более). При использовании команды LPM адрес, по которому производится чтение, определяется содержимым индексного регистра Z (см. далее). При этом старшие 15 битов содержимого регистра будут определять адрес слова (0…32К), а младший бит будет определять, какой из байтов будет прочитан: 0 — младший байт, 1 — старший байт (см. Рис. 1.5, а). Команда ELPM, в отличие от команды LPM, способна адресовать до 16 Мбайт памяти. При использовании этой команды адрес слова будет определяться содержимым регистра ввода/вывода RAMPZ совместно со старшими 15 битами содержимого регистра Z. Младший бит регистра Z будет по-прежнему определять, какой из байтов слова будет прочитан (см. Рис. 1.5, б). Понятно, что количество задействованных битов регистра RAMPZ зависит от объема памяти программ — в моделях с объемом памяти программ 128 Кбайт используется только младший бит RAMPZ0, а в моделях с 256 Кбайт памяти используются уже два младших бита — RAMPZ1 и RAMPZ0. Для обеспечения совместимости с будущими моделями микроконтроллеров при

записи значений в регистр RAMPZ незадействованные биты должны быть сброшены в 0.

Регистр RAMPZ расположен по адресу $3B ($5B) в основном пространстве регистров ввода/вывода микроконтроллеров ATmega128x и ATmega128xx/256xx, а его формат показан на Рис. 1.6.

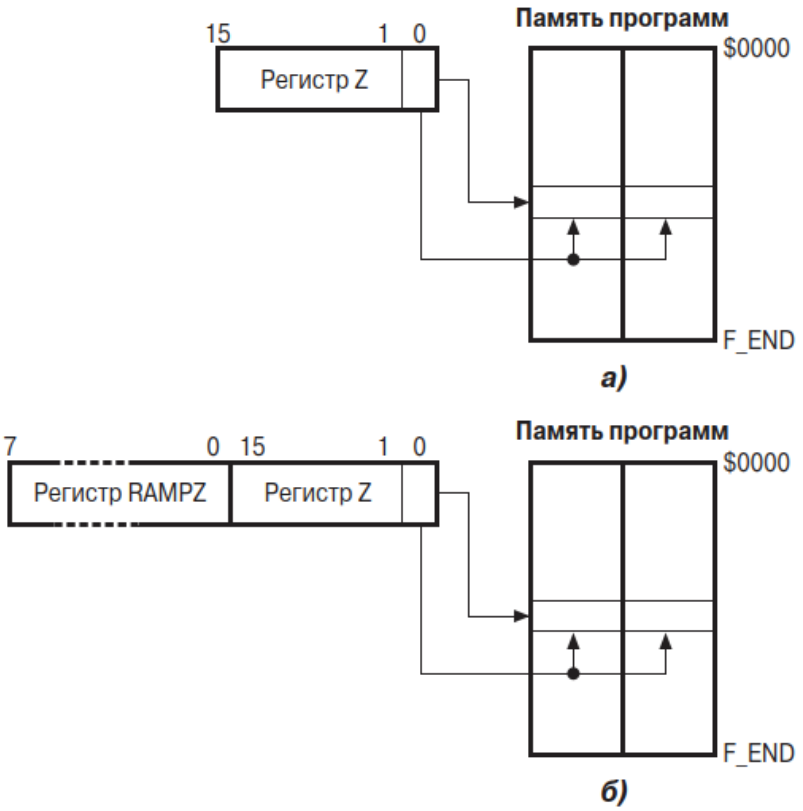


Рис.1.5. Косвенная адресация памяти программ при использовании команды LPM (а) и команды ELPM (б)

В заключение следует отметить, что FLASH-ПЗУ, используемое в микроконтроллерах AVR, рассчитано, как минимум, на 10 000 циклов стирания/записи (типовое значение — 100 000 циклов).

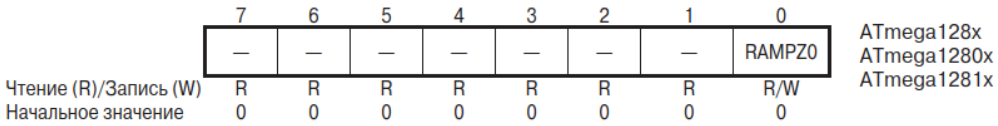


Рис.1.6 Формат регистра RAMPZ

Память данных

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

Регистровая память включает 32 регистра общего назначения (РОН), объединенных в файл, и служебные регистры ввода/вывода (РВВ). В слож- ных моделях с развитой периферией имеется также область дополнитель- ных

(extended) регистров ввода/вывода (ДРВВ). Под РВВ в памяти микроконтроллера отводится 64 байта, а под ДРВВ — 160 или 416 байт (в зависимости от модели).

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

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

Для долговременного хранения различной информации, которая мо- жет изменяться в процессе функционирования готовой системы (калибро- вочные константы, серийные номера, ключи и т. п.), в микроконтроллерах семейства может использоваться встроенная EEPROM-память. Ее объем составляет для различных моделей от 256 байт до 4 Кбайт. Эта память рас- положена в отдельном адресном пространстве, а доступ к ней осуществля- ется с помощью определенных РВВ.

Статическое ОЗУ

Прежде всего, следует сказать, что в микроконтроллерах AVR семейства Mega используется линейная организация памяти. Объем статического ОЗУ для различных моделей семейства составляет от 512 байт до 8 Кбайт (см. Табл. 1.1).

В адресном пространстве ОЗУ также расположены все регистры микроконтроллеров, под них отведены младшие 96 (256) адресов (см. Рис. 1.7). Остальные адреса отведены под 512/1К/2К/4К…64К ячеек статического ОЗУ.

Регистры общего назначения

Все регистры общего назначения объединены в регистровый файл быстрого доступа, структура которого показана на Рис. 1.8. Как было сказано выше, в микроконтроллерах AVR все 32 РОН непосредственно доступны АЛУ, в отличие от 8-битных микроконтроллеров других фирм, в кото- рых имеется только один такой регистр — рабочий регистр W (аккумулятор). Благодаря этому любой РОН может использоваться практически во всех командах и как операнд-источник, и как операнд-приемник. Такое ре- шение (в сочетании с конвейерной обработкой) позволяет АЛУ выполнять одну операцию (извлечение операндов из регистрового файла, выполнение команды и запись результата обратно в регистровый файл) за один такт.

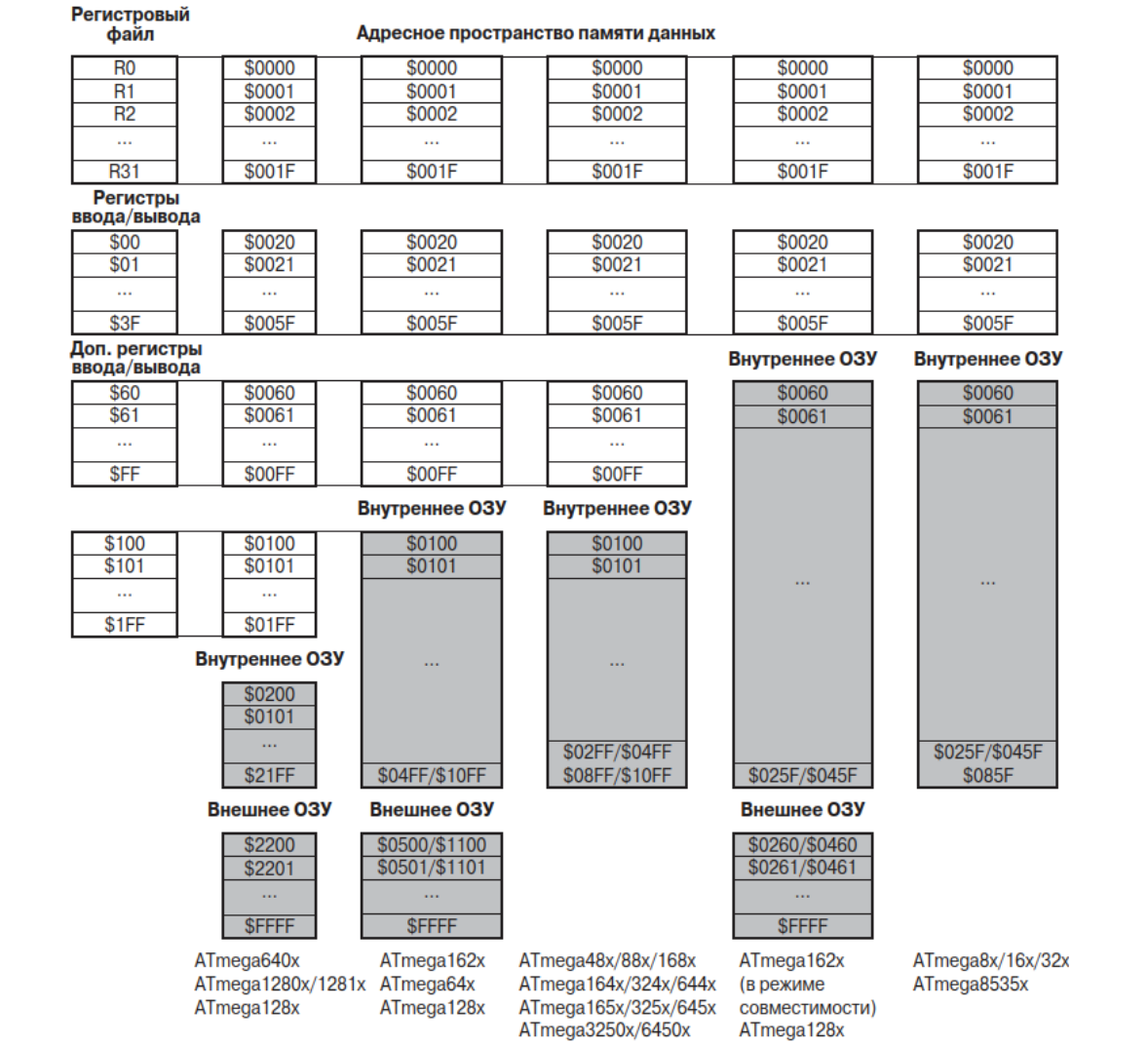
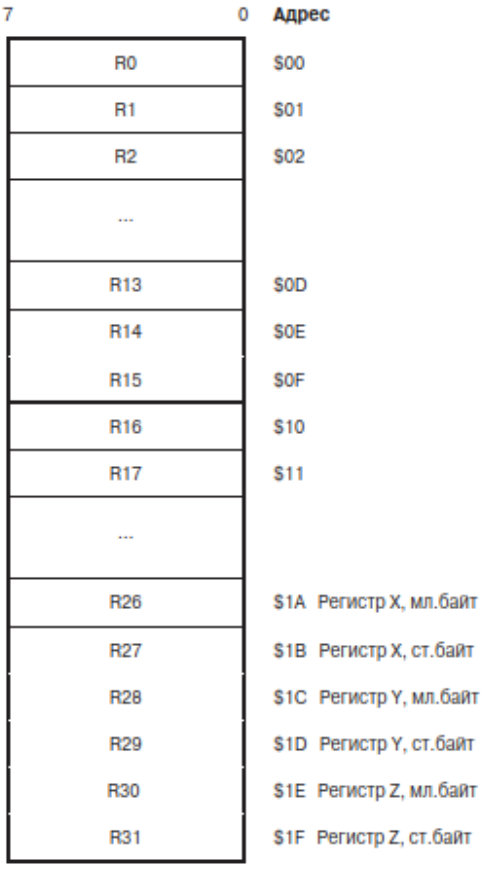


Рис.1.7. Организация статического



ОЗУ .

Рис.1.8. Структура регистрового файла

Последние 6 регистров файла (R26…R31) могут также объединяться в три 16-битных регистра X, Y и Z (см. **Рис. 1.9**), используемых в качестве указателей при косвенной адресации памяти данных. Как показано на **Рис. 1.7**, каждый регистр файла имеет свой собственный адрес в пространстве памяти данных. Поэтому к ним можно обращаться двумя способами — как к регистрам и как к памяти, несмотря на то, что физически эти регистры не являются ячейками ОЗУ. Такое решение является еще одной отличительной особенностью архитектуры AVR, повышающей эффективность работы микроконтроллера и его производительность.

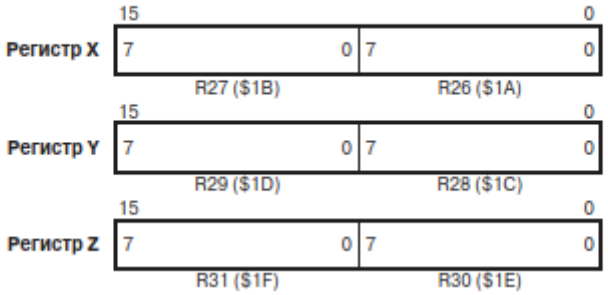


Рис.1.9. Регистры-указатели X, Y и Z

Регистры ввода/вывода

Все регистры ввода/вывода условно можно разделить на две группы: служебные регистры микроконтроллера и регистры, относящиеся к конкретным периферийным устройствам (в том числе регистры портов ввода/вывода).

Во всех микроконтроллерах AVR регистры ввода/вывода располагаются в так называемом пространстве ввода/вывода размером 64 байта. В большинстве моделей семейства Mega имеется также пространство дополнительных регистров ввода/вывода размером 160 или 416 байт. Введение дополнительных РВВ связано с тем, что для поддержки всех периферийных устройств, имеющихся в этих моделях, обычных 64-х РВВ недостаточно.

Распределение адресов пространства ввода/вывода (как основного, так и дополнительного) зависит от конкретной модели микроконтроллера или, если точнее, от состава и возможностей периферийных устройств данной модели. Размещение РВВ в пространстве ввода/вывода для всех моделей семейства приведено в Табл.1.3.

В таблицах и далее в книге при указании адресов РВВ в скобках указываются соответствующие им адреса ячеек ОЗУ. Соответственно, если адрес регистра указывается только в скобках, этот регистр расположен в пространстве дополнительных РВВ. Если адрес в таблице не указан, это означает, что для данной модели он зарезервирован, и запись по этому адресу запрещена (для совместимости с будущими моделями).

Таблица 1.3. **Регистры ввода/вывода моделей ATmega48x/88x/168x**

Название	Адрес	Функция
UDR0	($C6)	Регистр данных USART0

UBRR0H	($C5)	Регистр скорости передачи USART0, старший байт
UBRR0L	($C4)	Регистр скорости передачи USART0, младший байт
UCSR0C	($C2)	Регистр C управления и состояния USART0
UCSR0B	($C1)	Регистр B управления и состояния USART0
UCSR0A	($C0)	Регистр A управления и состояния USART0
TWAMR	($BD)	Регистр маски адреса TWI
TWCR	($BC)	Регистр управления TWI
TWDR	($BB)	Регистр данных TWI
TWAR	($BA)	Регистр адреса TWI
TWSR	($B9)	Регистр состояния TWI
TWBR	($B8)	Регистр скорости передачи TWI
ASSR	($B6)	Регистр состояния асинхронного режима
OCR2B	($B4)	Регистр B совпадения таймера/счетчика T2
OCR2A	($B3)	Регистр A совпадения таймера/счетчика T2
TCNT2	($B2)	Счетный регистр таймера/счетчика T2
TCCR2B	($B1)	Регистр B управления таймера/счетчика T2
TCCR2A	($B0)	Регистр A управления таймера/счетчика T2
OCR1BH	($8B)	Регистр B совпадения таймера/счетчика T1, старший байт
OCR1BL	($8A)	Регистр B совпадения таймера/счетчика T1, младший байт
OCR1AH	($89)	Регистр A совпадения таймера/счетчика T1, старший байт
OCR1AL	($88)	Регистр A совпадения таймера/счетчика T1, младший байт
ICR1H	($87)	Регистр захвата таймера/счетчика T1, старший байт
ICR1L	($86)	Регистр захвата таймера/счетчика T1, младший байт
TCNT1H	($85)	Счетный регистр таймера/счетчика T1, старший байт
TCNT1L	($84)	Счетный регистр таймера/счетчика T1, младший байт
TCCR1C	($82)	Регистр C управления таймера/счетчика T1
TCCR1B	($81)	Регистр B управления таймера/счетчика T1
TCCR1A	($80)	Регистр A управления таймера/счетчика T1
DIDR1	($7F)	Регистр 1 отключения цифровых входов
DIDR0	($7E)	Регистр 0 отключения цифровых входов
ADMUX	($7C)	Регистр управления мультиплексором АЦП
ADCSRB	($7B)	Регистр B управления и состояния АЦП
ADCSRA	($7A)	Регистр A управления и состояния АЦП
ADCH	($79)	Регистр данных АЦП, старший байт
ADCL	($78)	Регистр данных АЦП, младший байт
TIMSK2	($70)	Регистр маски прерываний от таймера/счетчика T2
TIMSK1	($6F)	Регистр маски прерываний от таймера/счетчика T1
TIMSK0	($6E)	Регистр маски прерываний от таймера/счетчика T0
PCMSK2	($6D)	Регистр маски 2-го прерывания по изменению состояний выводов
PCMSK1	($6C)	Регистр маски 1-го прерывания по изменению состояний выводов
PCMSK0	($6B)	Регистр маски 0-го прерывания по изменению состояний выводов
EICRA	($69)	Регистр A управления внешними прерываниями

PCICR	($68)	Регистр управления прерываниями по изменению состояний выводов
OSCCAL	($66)	Регистр калибровки тактового генератора
PRR	($64)	Регистр снижения потребляемой мощности
CLKPR	($61)	Регистр предделителя тактового сигнала
WDTCSR	($60)	Регистр управления и состояния сторожевого таймера
SREG	$3F ($5F)	Регистр состояния
SPH	$3E ($5E)	Указатель стека, старший байт
SPL	$3D ($5D)	Указатель стека, младший байт
SPMCSR	$37 ($57)	Регистр управления и состояния операций записи в память программ
MCUCR	$35 ($55)	Регистр управления микроконтроллера
MCUSR	$34 ($54)	Регистр состояния микроконтроллера
SMCR	$33 ($53)	Регистр управления спящим режимом
ACSR	$30 ($50)	Регистр управления и состояния аналогового компаратора
SPDR	$2E ($4E)	Регистр данных SPI
SPSR	$2D ($4D)	Регистр состояния SPI
SPCR	$2C ($4C)	Регистр управления SPI
GPIOR2	$2B ($4B)	2-й регистр ввода/вывода общего назначения
GPIOR1	$2A ($4A)	1-й регистр ввода/вывода общего назначения
OCR0B	$28 ($48)	Регистр B совпадения таймера/счетчика T0
OCR0A	$27 ($47)	Регистр A совпадения таймера/счетчика T0
TCNT0	$26 ($46)	Счетный регистр таймера/счетчика T0
TCCR0B	$25 ($45)	Регистр B управления таймера/счетчика T0
TCCR0A	$24 ($44)	Регистр A управления таймера/счетчика T0
GTCCR	$23 ($43)	Общий регистр управления таймеров/счетчиков
EEARH	$22 ($42)	Регистр адреса EEPROM, старший байт
EEARL	$21 ($41)	Регистр адреса EEPROM, младший байт
EEDR	$20 ($40)	Регистр данных EEPROM
EECR	$1F ($3F)	Регистр управления EEPROM
GPIOR0	$1E ($3E)	0-й регистр ввода/вывода общего назначения
EIMSK	$1D ($3D)	Регистр маски внешних прерываний
EIFR	$1C ($3C)	Регистр флагов внешних прерываний
PCIFR	$1B ($3B)	Регистр флагов прерываний по изменению состояний выводов
TIFR2	$17 ($37)	Регистр флагов прерываний от таймера/счетчика T2
TIFR1	$16 ($36)	Регистр флагов прерываний от таймера/счетчика T1
TIFR0	$15 ($35)	Регистр флагов прерываний от таймера/счетчика T0
PORTD	$0B ($2B)	Регистр данных порта D
DDRD	$0A ($2A)	Регистр направления данных порта D
PIND	$09 ($29)	Выводы порта D
PORTC	$08 ($28)	Регистр данных порта C
DDRC	$07 ($27)	Регистр направления данных порта C
PINC	$06 ($26)	Выводы порта C
PORTB	$05 ($25)	Регистр данных порта B
DDRB	$04 ($24)	Регистр направления данных порта B
PINB	$03 ($23)	Выводы порта B

К регистрам ввода/вывода, расположенным в основном пространстве ввода/вывода, можно напрямую обратиться с помощью команд IN и OUT, выполняющих пересылку данных между одним из 32-х РОН и простран- ством ввода/вывода. В системе команд имеется также четыре команды по- битового доступа, использующие в качестве операндов регистры вво- да/вывода: команды установки/сброса отдельного бита (SBI и CBI) и ко- манды проверки состояния отдельного бита (SBIS и SBIC). К сожалению, эти команды могут обращаться только к 1-й половине основного про- странства ввода/вывода (адреса $00…$1F).

Помимо непосредственной адресации (с помощью команд IN и OUT), к РВВ можно обращаться и как к ячейкам ОЗУ с помощью соответствую- щих команд ST/SD/SDD и LD/LDS/LDD (для дополнительных РВВ этот способ является единственно возможным). В первом случае используются адреса РВВ, принадлежащие основному пространству ввода/вывода ($00…$3F). Во втором случае адрес РВВ необходимо увеличить на $20.

Среди РВВ есть один регистр, используемый наиболее часто в процессе выполнения программ. Это регистр состояния SREG. Он располагается по адресу $3F ($5F) и содержит набор флагов, показывающих текущее со- стояние микроконтроллера. Большинство флагов автоматически устанавливаются в 1 или сбрасываются в 0 при наступлении определенных событий (в соответствии с результатом выполнения команд). Все биты этого регистра доступны как для чтения, так и для записи; после сброса микроконтроллера все биты регистра сбрасываются в 0. Формат этого регистра показан на Рис. 1.10, а его описание приведено в Табл. 1.4.

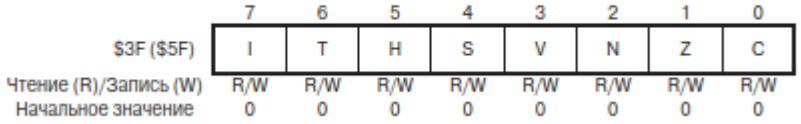


Рис.1.10. Формат регистра состояния SREG

Кроме того, в последних моделях микроконтроллеров семейства поя- вились 3 регистра ввода/вывода общего назначения — GPIOR0, GPIOR1 и GPIOR2. В этих регистрах можно хранить любую информацию, однако ос- новное их назначение — сохранение глобальных переменных и регистра SREG. Регистры GPIOR0…2 располагаются в младшей половине основно- го пространства ввода/вывода и соответственно могут использоваться в командах побитового доступа SBI/CBI и SBIS/SBIC.

Таблица 1.4. **Биты регистра состояния SREG**

Бит	Название	Описание
7	I	**Общее разрешение прерываний.** Для разрешения прерываний этот флаг должен быть установлен в 1. Разрешение/запрещение от- дельных прерываний производится установкой

		или сбросом соответствующих битов регистров масок прерываний (регистров управления прерываниями). Если флаг сброшен, то прерывания за- прещены независимо от состояния битов этих регистров. Флаг сбрасывается аппаратно после входа в прерывание и восста- навливается командой RETI для разрешения обработки следующих прерываний
6	T	**Хранение копируемого бита.** Этот бит регистра используется в качест- ве источника или приемника командами копирования битов BLD (Bit LoaD) и BST (Bit STore). Заданный бит любого РОН может быть скопирован в этот бит командой BST или установлен в соответствии с содержимым данного бита командой BLD
5	H	**Флаг половинного переноса.** Этот флаг устанавливается в 1, если произошел перенос из младшей половины байта (из 3-го бита в 4-й) или заем из старшей половины байта при выполнении неко- торых арифметических операций
4	S	**Флаг знака.** Этот флаг равен результату операции «Исключающее ИЛИ» (XOR) между флагами N (отрицательный результат) и V (переполнение числа в дополнительном коде). Соответственно, этот флаг устанавливается в 1, если результат выполнения арифметической операции меньше нуля
3	V	**Флаг переполнения дополнительного кода.** Этот флаг устанавливается в 1 при переполнении разрядной сетки знакового результата. Используется при работе со знаковыми числами (представленными в дополнительном коде). Подробнее — см. описание системы команд
2	N	**Флаг отрицательного значения.** Этот флаг устанавливается в 1, если старший бит (7-й) результата операции равен 1. В противном случае флаг равен 0
1	Z	**Флаг нуля.** Этот флаг устанавливается в 1, если результат выполне- ния операции равен нулю

0	C	**Флаг переноса.** Этот флаг устанавливается в 1, если в результате выполнения операции произошел выход за границы байта

Использование внешнего ОЗУ

Микроконтроллеры ATmega8515x, ATmega162x, ATmega64x/128x и ATmega640x/1280x/1281x/2560x/2561x имеют возможность подключения внешнего статического ОЗУ объемом до 64 Кбайтнизмы через цифро-аналоговые преобразователи и усилите- ли) или во внешнюю память (на диски, ленты, флэш-карты) или в телекомму-никационные сети.

Выводы микроконтроллеров, используемые для подключения внеш- него ОЗУ, сведены в **Табл. 1.5**. Во всех моделях эти выводы являются ли- ниями портов ввода/вывода общего назначения. При включенном интер- фейсе внешнего ОЗУ режим работы этих выводов определяется не со- держимым регистров направления передачи данных, а самим микроконтроллером.

Таблица 1.5. **Выводы, используемые для подключения внешнего ОЗУ**

азвание	**ATmega8515x, ATmega162x**	**ATmega64x/128x, ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x**	**Описание**
D0…AD7	PA0…PA7 (PORTA)		Мультиплексированная ША/ШД
8…A15	PC0…PC7 (PORTC)		Старший байт ША
LE	PE1	PG2	Строб адреса
$\overline{D}$	PD7	PG1	Строб записи
$\overline{R}$	PD6	PG0	Строб чтения

При отсутствии обращения к внешней памяти выводы порта A переводятся микроконтроллером в третье состояние. Этого можно избежать, если подключить к выходам порта внутренние подтягивающие резисторы либо установить в 1 бит XMBK регистра SFIOR (модели ATmega8515x и ATmega162x) или XMCRB (остальные модели). При установленном бите на выводах порта A всегда сохраняется последнее выведенное значение.

Подключение внешнего ОЗУ к микроконтроллеру показано на **Рис. 1.11**. Как видно из рисунка, для этого дополнительно потребуется ре- гистр-защелка. В качестве такой защелки, как правило, используют мик- росхему типа 74x573 или аналогичную, в которой защелкивание данных происходит по НИЗКОМУ уровню управляющего сигнала.При тактовой частоте (более 8 МГц при VCC = 4 В и более 4 МГц при VCC = 2.7 В) рекомендуется использовать быстродействующие микросхемы, такие как SN74AHC573.

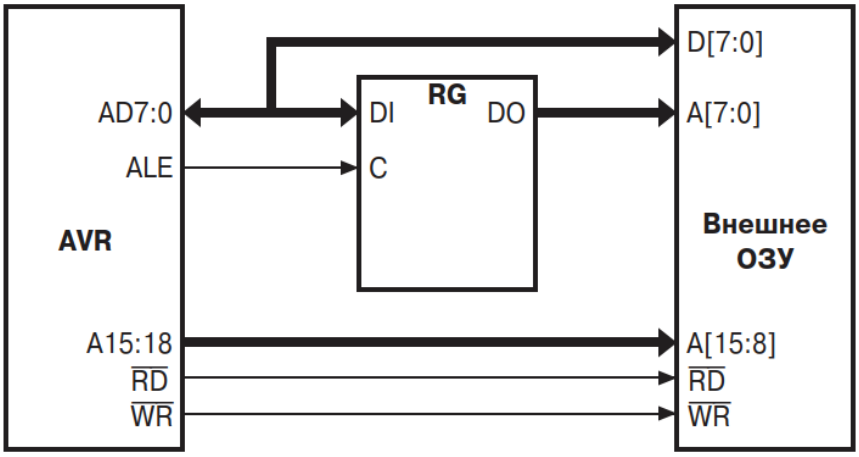


Рис.1.11. Подключение внешнего ОЗУ
к микроконтроллеру

Все микроконтроллеры семейства, допускающие подключение внеш- него ОЗУ, имеют следующие возможности по работе с внешней памятью:

- управление длительностью цикла обращения к внешней памяти;
- разбиение внешней памяти на два сектора с возможностью задания различной длительности цикла обращения для каждого сектора;
- управление разрядностью шины адреса;
- удержание значений на шине данных для уменьшения потребляемо- го тока.

Для управления описанными возможностями используются два или три регистра (в зависимости от модели), которые перечислены в **Табл. 1.6**.

Таблица 1.6. **Регистры для управления внешней памятью**

Название	Описание	Адрес	Модель
MCUCR	Регистр управления микроконтроллера	$35 ($55)	ATmega8515x, ATmega162x
EMCUCR	Дополнительный регистр управления микро- контроллера	$36 ($56)	
SFIOR	Регистр специальных функций	$30 ($50)	

MCUCR	Регистр управления микроконтроллера	$35 ($55)	ATmega64x/128x
XMCRA	Регистр А управления внешней памятью	($6D)	
XMCRB	Регистр В управления внешней памятью	($6C)	
XMCRA	Регистр А управления внешней памятью	($74)	ATmega650, ATmega1280x/1281x, ATmega2560x/2561x
XMCRB	Регистр В управления внешней памятью	($75)	

Формат регистра MCUCR приведен на Рис. 1.12. Для работы с внешней памятью в нем используются только два бита (Табл.1.7).

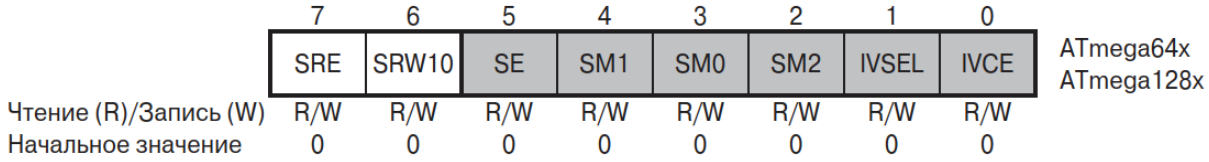


Рис.1.12. Формат регистра MCUCR

Таблица 1.7. **Биты регистра MCUCR при управлении внешней памятью**

Бит	Название	Описание
7	SRE	**Разрешение работы с внешней памятью.** Установка этого бита в 1 разрешает работу с внешней памятью. Установки регистров направления передачи данных для соответствующих выводов (см. **Табл. 2.13**) при этом игнорируются. При сброшенном бите SRE обращение к внешней памяти запрещено, а выводы используются как линии ввода/вывода общего назначения
6	SRW10	**Выбор числа тактов ожидания (верхний сектор).** Этот бит является младшим битом селектора длительности обращения ко второму (верхнему) сектору памяти

Формат регистров XMCRA и EMCUCR приведен на **Рис. 1.13**. В этих регистрах для работы с внешней памятью используются только 6 битов. Однако в регистре EMCUCR два оставшихся бита используются для дру- гих целей, а в регистре XMCRA они не используются вообще. Описание
используемых для управления внешней памятью битов этих регистров приведено

в **Табл. 1.8** и **Табл. 1.9**. Обратите внимание на то, что в моде- лях ATmega640x/128xx/256xx биты SRWxx регистра XMCRA сгруппирова- ны совершенно иначе, нежели в остальных моделях.

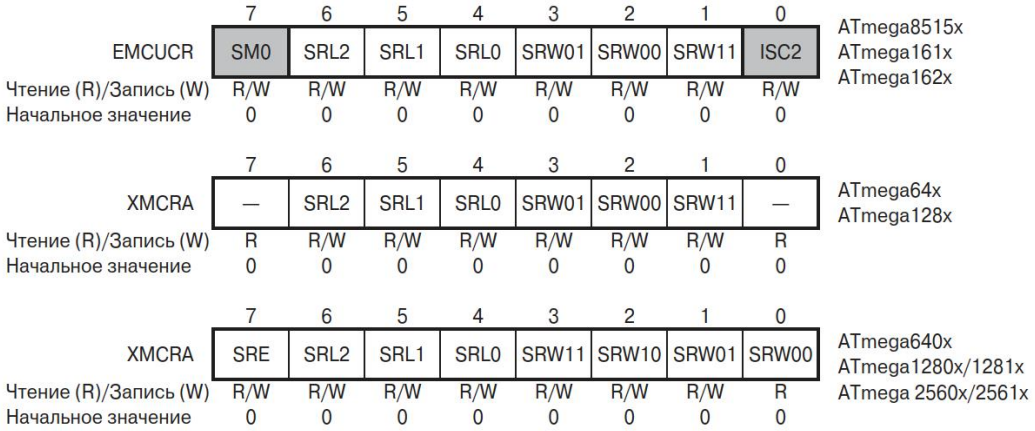


Рис.1.13. Формат регистров XMCRA и EMCUCR

Таблица 1.8. **Биты регистров XMCRA (модели ATmega64x/128x) и EMCUCR**

Бит	Название	Описание
7	—	Зарезервирован, читается как 0
6	SRL2	**Задание границ секторов.** Значение этих битов определяет положе- ние границы между двумя секторами внешней памяти
5	SRL1	
4	SRL0	
3	SRW01	**Выбор числа тактов ожидания (нижний сектор).** Эти биты являются селектором длительности обращения к первому (нижнему) сек- тору внешней памяти
2	SRW00	
1	SRW11	**Выбор числа тактов ожидания (верхний сектор).** Этот бит является старшим битом селектора длительности обращения ко второму (верхнему) сектору памяти (младший бит — в регистре MCUCR)
0	—	Зарезервирован, читается как 0

Таблица 1.9. **Биты регистров XMCRA (модели ATmega640x/128xx/256xx)**

Бит	Название	Описание
7	SRE	**Разрешение работы с внешней памятью.** Установка этого бита в 1 разрешает работу с внешней памятью. Установки регистров на- правления передачи данных для соответствующих выводов (см. **Табл. 2.13**) при этом игнорируются. При сброшенном бите SRE обращение к внешней памяти запрещено, а выводы используются как линии ввода/вывода общего назначения

6	SRL2	**Задание границ секторов.** Значение этих битов определяет положе- ние границы между двумя секторами внешней памяти
5	SRL1	
4	SRL0	
3	SRW11	**Выбор числа тактов ожидания (верхний сектор).** Эти биты являются селектором длительности обращения ко второму (верхнему) сек- тору внешней памяти
2	SRW10	
1	SRW01	**Выбор числа тактов ожидания (нижний сектор).** Этот бит является старшим битом селектора длительности обращения к первому (нижнему) сектору памяти
0	SRW00	

Формат регистров XMCRB и SFIOR приведен на **Рис. 1.14**. В этих ре- гистрах для работы с внешней памятью используются только 4 бита. Как и в регистре EMCUCR, в регистре SFIOR остальные биты используются для других целей. Описание используемых для управления внешней памятью битов регистров XMCRB и SFIOR приведено в **Табл. 1.10.**

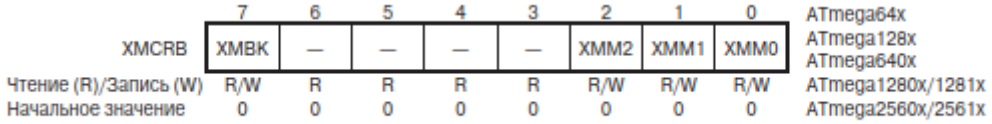


Рис.1.14. Формат регистров XMCRB и SFIOR

Таблица 1.10. **Биты регистров XMCRB и SFIOR**

Бит		**Название**	**Описание**
XM CRB	**SFIOR**		
7	6	XMBK	**Разрешение удержания значений на шине данных.** Если этот бит установлен в 1, то на выводах AD7…AD0 всегда удер- живается последнее выведенное значение (даже если микроконтроллер должен перевести их в третье состоя- ние). При сброшенном бите эта функция отключена. **Установка бита XMBK действует, даже если работа с внеш- ней памятью запрещена!**
2	5	XMM2	**Маска старшего байта шины адреса.** Содержимое этих би- тов определяет количество выводов порта C, задейство- ванных под
1	4	XMM1	
0	3	XMM0	

			шину адреса. Незадействованные выводы мо- гут использоваться как линии ввода/вывода общего назначения

Микроконтроллеры семейства Mega позволяют использовать микро- схемы внешнего ОЗУ с различным временем доступа. Подстройка под раз- личные микросхемы осуществляется изменением длительности цикла об- ращения к внешней памяти. Более того, имеется возможность разбить все адресное пространство внешнего ОЗУ на два сектора, для каждого из кото- рых может быть задана своя длительность цикла обращения. В общем виде конфигурация внешнего ОЗУ, подключенного к микроконтроллеру, пока- зана на **Рис. 1.15.**

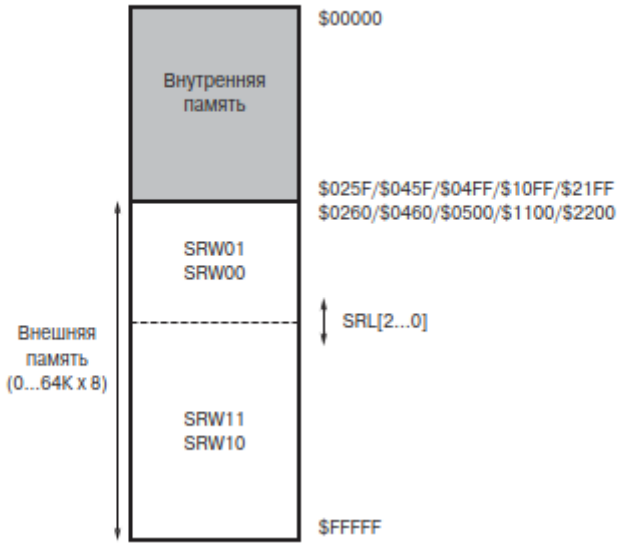


Рис.1.15. Конфигурация внешнего ОЗУ

Положение границы между секторами определяется содержимым битов SRL2…SRL0 согласно **Табл. 1.11**. По умолчанию эти биты сброшены в 0, т. е. вся область памяти состоит из одного сектора.

***Таблица 1.11.* Определение секторов внешней памяти**

SRL2	SRL1	SRL0	Нижняя граница 1-го сектора	
			ATmega8515x, ATmega162x, ATmega64x/128x	ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x
0	0	0	—	—
0	0	1	$1FFF	—
0	1	0	$3FFF	$3FFF
0	1	1	$5FFF	$5FFF
1	0	0	$7FFF	$7FFF
1	0	1	$9FFF	$9FFF
1	1	0	$BFFF	$BFFF
1	1	1	$DFFF	$DFFF

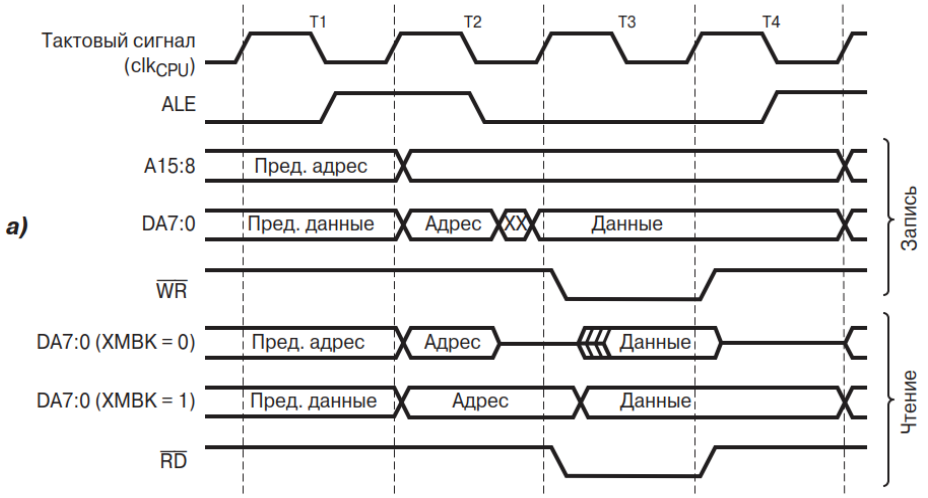
Изменение длительности цикла обращения к внешней памяти осуществляется заданием дополнительных тактов ожидания с помощью битов SRW01:SRW00 — для первого сектора и SRW11:SRW10 — для второго (**Табл. 1.12**).

Таблица 1.12. **Определение числа тактов ожидания**

SRW*n*1[1)]	**SRW*n*0**[1)]	**Такты ожидания**
0	0	Нет тактов ожидания
0	1	Один такт ожидания во время действия строба чтения/записи
1	0	Два такта ожидания во время действия строба чтения/записи
1	1	Два такта ожидания во время действия строба чтения/записи и один такт перед выставлением на шину нового адреса
[1)] *n* = 0 (нижний) или 1 (верхний) сектор.		

Временные диаграммы обращения к внешнему ОЗУ при различных установках битов SRW*n*1:SRW*n*0 приведены на **Рис. 1.16**.

Как уже было сказано, при работе с внешним ОЗУ используется 16-разрядная шина адреса, позволяющая адресовать 64К адресов. Однако для многих приложений не требуется такого большого объема внешней памяти. Специально для таких случаев все микроконтроллеры (за исключением модели ATmega162x в режиме совместимости с ATmega161x) позволяют уменьшить разрядность шины адреса.



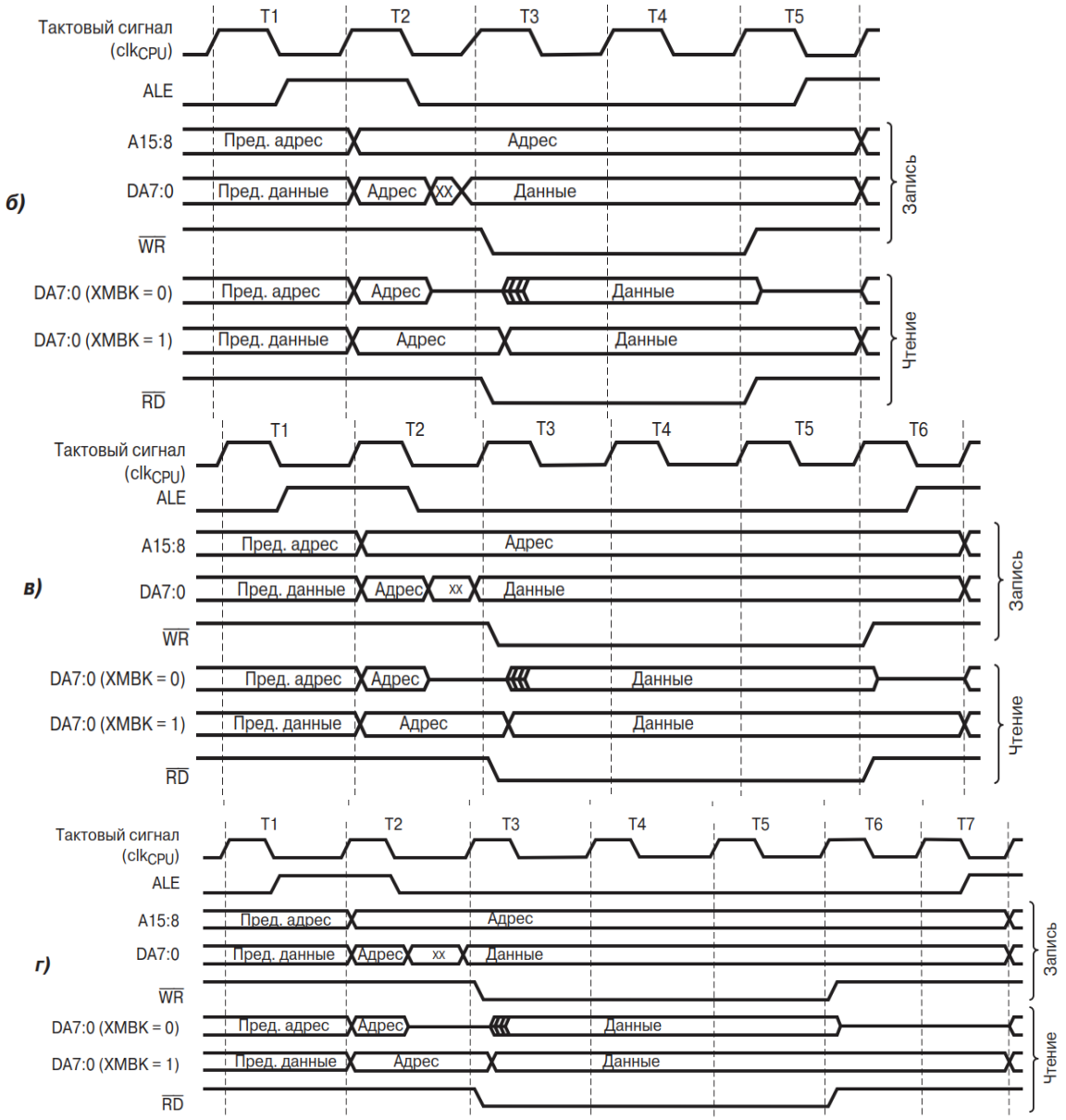


Рис.1.16. Временные диаграммы обращения к внешнему ОЗУ: SRW*n*1:SRW*n*0 = 00 (а); = 01 (б); = 10 (в); = 11 (г)

Как было показано на **Рис. 1.15**, младшие адреса внешней памяти отоб- ражаются на адресное пространство внутреннего ОЗУ, в результате чего часть емкости внешнего ОЗУ остается незадействованной. Тем не менее, имеется возможность полного использования емкости микросхем внешне- го ОЗУ. Если объем внешнего ОЗУ меньше 64 Кбайт, то для доступа к млад- шим адресам внешнего ОЗУ достаточно просто использовать адреса, лежа- щие вне области физических адресов внешнего ОЗУ. Поскольку старшие биты шины адреса не используются, то при обращении по этим адресам бу- дут адресоваться ячейки внешнего ОЗУ с младшими адресами, начиная с 0x0000. Возьмем, к примеру, микроконтроллер ATmega128x с внешним ОЗУ объемом 32 Кбайт (см. **Рис. 1.17**). В этом случае при обращении по адресам 0x8000…0x90FF мы будем в действительности обращаться к ячейкам внешнего ОЗУ с адресами 0x0000…0x10FF. Таким образом, в распоряжении прикладной программы окажется линейная область памяти размером 32 Кбайт с адресами 0x1100…0x90FF.

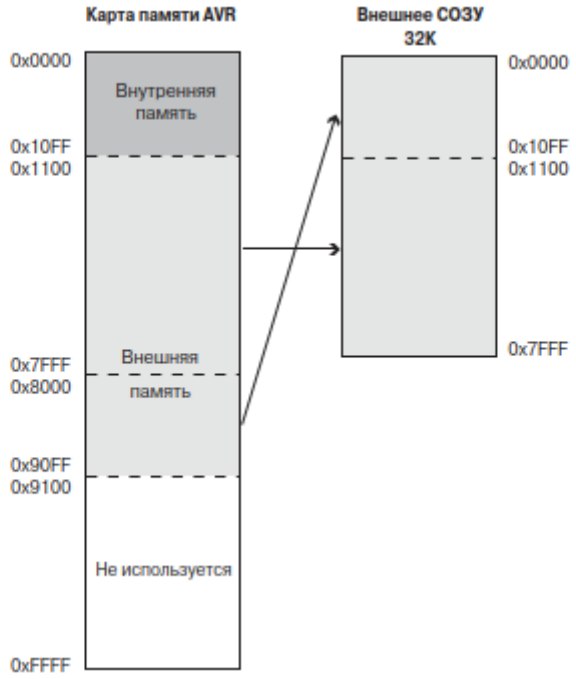


Рис.1.17. Полное использование емкости микросхемы ОЗУ (32 Кбайт) на примере микроконтроллера ATmega128x

Второй способ полного использования емкости микросхем внешнего ОЗУ основан на программном управлении старшими битами адреса. Для выполнения этого «трюка» порт С должен быть настроен на вывод, а в защелке порта должно быть записано $00. При маскировании старших битов адреса (с помощью битов XMM2…XMM0 регистра XMCRB) на соответствующие выводы порта С будут выставлены нули и произойдет обращение к младшим адресам внешнего ОЗУ, начиная с адреса $0000. Ниже при- ведены два примера программной реализации описанных действий (для микроконтроллера ATmega128x).

Пример на ассемблере

```
; Константа OFFSET определена как 0x2000 для гарантированного
; обращения к внешней памяти ldi    r16, 0xFF
out     DDRC, r16  ; Все выводы порта С — выходы ldir16, 0x00
out     PORTC, r16; Записали в защелку порта 0x00 ldi r16,
(1<<XMM1)|(1<<XMM0)
out     XMCRB, r16        ; Высвободили биты PC7…5 ldi       r16, 0xAA
sts     0x0001+OFFSET, r16     ; Записали 0xAA по адресу 0x0001 внешней памяти
ldi     r16, (0<<XMM1)|(0<<XMM0)
out     XMCRB, r16        ; Переназначили биты PC7…5 ldi     r16, 0x55
sts     0x0001+OFFSET, r16     ; Записали 0x55 по адресу (OFFSET+1)
; внешней памяти
```

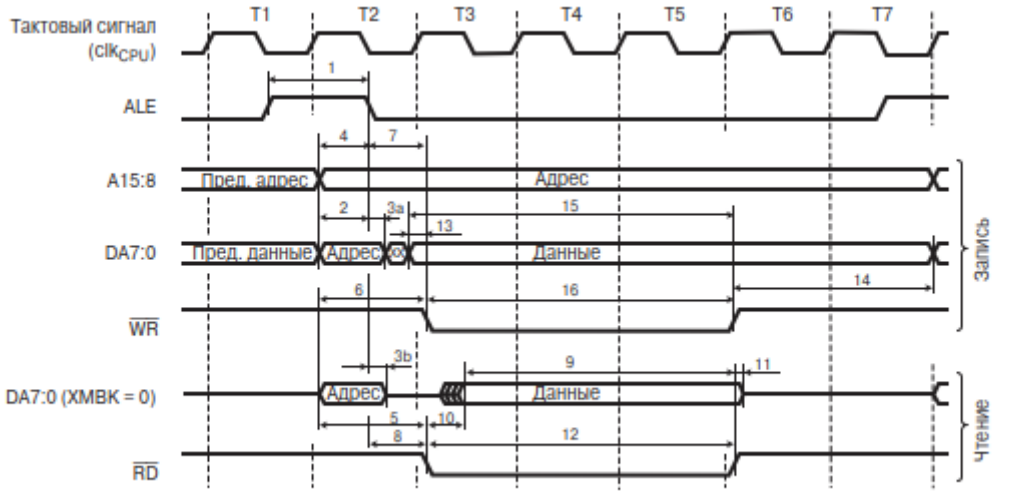
Пример на Си

```
#define OFFSET 0x2000 void XRAM_example(void)
{
unsigned char *p = (unsigned char *) (OFFSET + 1); DDRC = 0xFF;
PORTC = 0x00;
XMCRB = (1<<XMM1) | (1<<XMM0);
*p = 0xaa; XMCRB = 0x00;
*p = 0x55;
```

}

Необходимо помнить, что по сравнению с обращением к внутреннему ОЗУ обращение к внешнему ОЗУ требует, в зависимости от режима, на 1, 2, 3 или 4 такта больше для каждого байта, обрабатываемого командой. Та- ким образом, время выполнения команд передачи данных (LD, ST, LDS, STS, PUSH и POP) увеличивается на 1, 2, 3 или 4 такта. Если стек расположен во внешнем ОЗУ, то время перехода к обработке прерываний, вызова и возврата из подпрограмм увеличивается на 2, 4, 6, 8, а для моделей ATmega2560x/2561x — на 3, 6, 9 и 12 тактов. Это связано с тем, что во время выполнения указанных операций происходит сохранение или восстановление 16/17-битного счетчика команд и, кроме того, при работе с внешней памятью не используется конвейеризация.

Полные временные диаграммы обращения к внешнему ОЗУ с указанием всех параметров сигналов приведены на Рис. 1.18. Значения этих параметров приведены в Табл. 1.13.



Примечание. Такты T5...T7 присутствуют только при использовании дополнительных тактов ожидания

Рис.1.18. Временные диаграммы обращения к внешнему ОЗУ

Таблица 1.13. **Динамические параметры сигналов при обращении к внешнему ОЗУ**

№	Обозначение	Параметр	V_{CC} = 4.5…5.5 В		V_{CC} = 2.7…5.5 В	Ед. изм.
			min	max	max	
0	$1/t_{CLCL}$	Тактовая частота	0	16	8	МГц
1	t_{LHLL}	Длительность сигнала ALE	$1.0t_{CLCL}$ – 10	—	—	нс
2	t_{AVL}	Время установления сиг-	$0.5t_{CLCL}$ –	—	—	нс

	L	нала ALE относительно сигналов адреса A[7...0]	5			
3a	t_{LLAX_ST}	Время сохранения сигна- лов адреса A[7…0] после сигнала ALE (команды ST/STD/STS)	5	—	—	нс
3b	t_{LLAX_LD}	Время сохранения сигна- лов адреса A[7…0] после сигнала ALE (команды LD/LDD/LDS)	5	—	—	нс
4	t_{AVLLC}	Время установления сиг- нала ALE относительно сигналов адреса A[15…8]	$0.5t_{CLCL} - 5$	—	—	нс
5	t_{AVRL}	Время установления сигна- ла RD относительно адреса	$1.0t_{CLCL} - 10$	—	—	нс

1.3. Способы адресации памяти данных

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

Вообще говоря, в действительности способов адресации всего два: прямая адресация и косвенная. Однако каждый способ адресации имеет несколько разновидностей в зависимости от того, к какой области памяти производится обращение (при прямой адресации) или какие дополнитель- ные действия выполняются над индексным регистром (при косвенной ад- ресации).

На рисунках этого подраздела, а также далее в книге встречается аббревиатура «КОП». Эта аббревиатура обозначает часть (или части) слова команды, содержащую значение кода операции.

Прямая адресация

При прямой адресации адреса операндов содержатся непосредственно в слове команды. В соответствии со структурой памяти данных существу- ют следующие разновидности прямой адресации: прямая адресация одно- го РОН, прямая адресация двух РОН, прямая адресация РВВ, прямая адре- сация ОЗУ.

Прямая адресация одного регистра общего назначения

Этот способ адресации используется в командах, оперирующих с од- ним из регистров общего назначения. При этом адрес регистра-операнда (его номер) содержится в пяти битах слова команды (**Рис. 1.19**). Положе- ние разрядов d на рисунке показано условно.

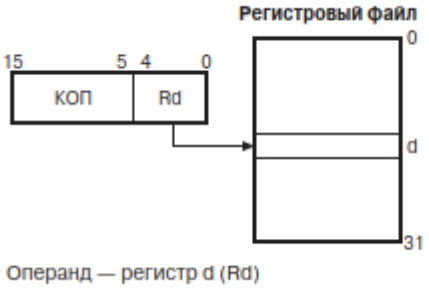


Рис.1.19. Прямая адресация одного регистра общего назначения

Примером команд, использующих этот способ адресации, являются команды работы со стеком (PUSH, POP), команды инкрементирования (INC), декрементирования (DEC), а также некоторые команды арифмети- ческих операций.

Прямая адресация двух регистров общего назначения

Этот способ адресации используется в командах, оперирующих одно- временно с двумя регистрами общего назначения. При этом адрес регист- ра-источника содержится в битах 9, 3…0 (5 битов), а адрес регистра-при- емника — в битах 8…4 (5 битов) слова команды (**Рис. 1.20**). Положение битов r и d на рисунке показано условно.

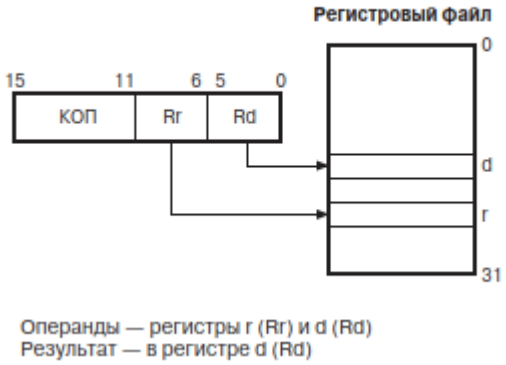


Рис.1.20. Прямая адресация двух регистров общего назначения

К командам, использующим этот способ адресации, относятся коман- да пересылки данных из регистра в регистр (MOV), а также большинство команд арифметических операций.

Здесь необходимо сделать одно замечание. Дело в том, что некоторые команды, имеющие только один регистр-операнд, тем не менее использу- ют рассматриваемый способ адресации. Просто в этом случае источником и приемником является один и тот же регистр. В качестве примера можно привести команду очистки регистра (CLR Rd), которая в действительнос- ти выполняет операцию «Исключающее ИЛИ» регистра с самим собой (EOR Rd, Rd).

Прямая адресация регистра ввода/вывода

Данный способ адресации используется командами пересылки данных между регистром ввода/вывода, расположенным в основном пространстве ввода/вывода, и регистровым файлом — IN и OUT. В этом случае адрес регистра ввода/вывода содержится в битах 10, 9, 3…0 (6 битов), а адрес РОН — в битах 8…4 (5 битов) слова команды (Рис. 1.21). Положение битов r/d и P на рисунке показано условно.

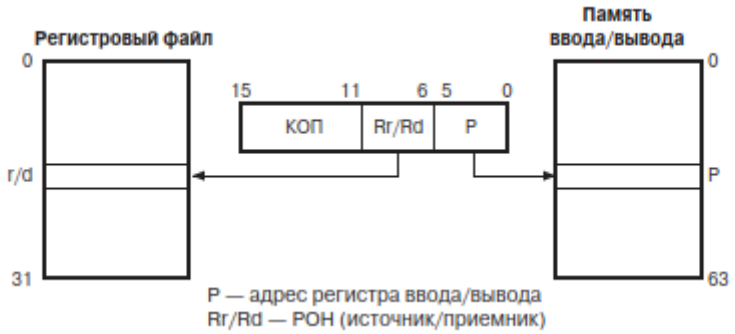


Рис.1.21. Прямая адресация регистра ввода/вывода

Прямая адресация ОЗУ

Данный способ используется для обращения ко всему адресному про- странству памяти данных.

В системе команд микроконтроллеров семейства имеется только две команды, использующие этот способ адресации. Это команды пересылки байта между одним из РОН и ячейкой ОЗУ — LDS и STS. Каждая из этих команд занимает в памяти программ два слова (32 бита). В первом слове содержится код операции и адрес регистра общего назначения (в битах с 8-го по 4-й). Во

втором слове находится адрес ячейки памяти, к которой происходит обращение **(Рис. 1.22).**

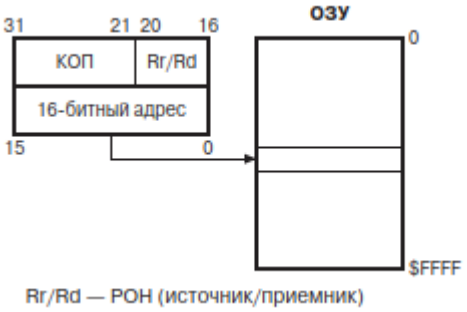


Рис.1.22. Прямая адресация ОЗУ

Косвенная адресация

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

Простая косвенная адресация

При использовании команд простой косвенной адресации обращение производится к ячейке памяти, адрес которой находится в индексном ре- гистре (Рис. 1.23). Никаких действий с содержимым индексного регистра при этом не производится.

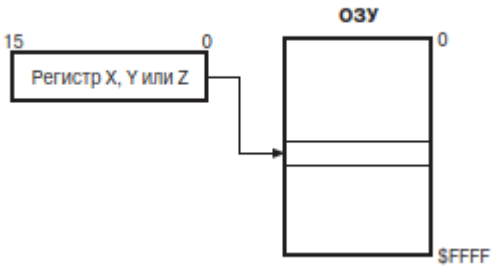


Рис.1.23. Простая косвенная адресация

Микроконтроллеры поддерживают 6 команд (по 2 для каждого индексного регистра) простой косвенной адресации: LD Rd,X/Y/Z (пересылка байта из ОЗУ в РОН) и ST X/Y/Z,Rd (пересылка байта из РОН в ОЗУ). Ад- рес регистра общего назначения содержится в битах 8…4 слова команды.

Относительная косвенная адресация

При использовании команд относительной косвенной адресации ад- рес ячейки памяти, к которой производится обращение, получается суммированием содержимого индексного регистра (Y или Z) и константой, задаваемой в команде. Другими словами, производится обращение по ад- ресу, указанному в команде, относительно адреса, находящегося в индекс- ном регистре. Иллюстрация данного способа адресации приведена на **Рис. 1.24**.

Положение битов n и q на рисунке показано условно.

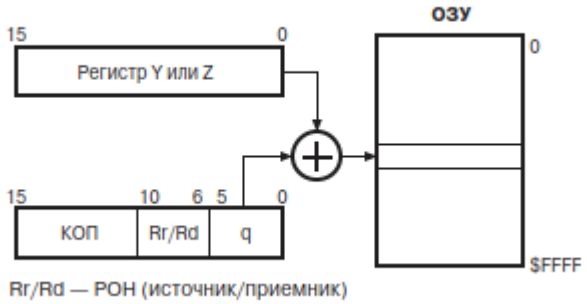


Рис.1.24. Относительная косвенная адресация

Микроконтроллеры семейства Mega поддерживают 4 команды относительной косвенной адресации (две — для регистра Y и две — для регистра Z): LDD Rd, Y+q/Z+q (пересылка байта из ОЗУ в РОН) и ST Y+q/Z+q, Rr (пересылка байта из РОН в ОЗУ). Адрес регистра общего назначения содержится в битах 8…4 слова команды, а величина смещения — в битах 13, 11, 10, 2…0. Поскольку под значение смещения отводится только 6 битов, оно не может превышать 64.

Косвенная адресация с преддекрементом

При использовании команд косвенной адресации с преддекрементом содержимое индексного регистра сначала уменьшается на 1, а затем производится обращение по полученному адресу (**Рис. 1.25**).

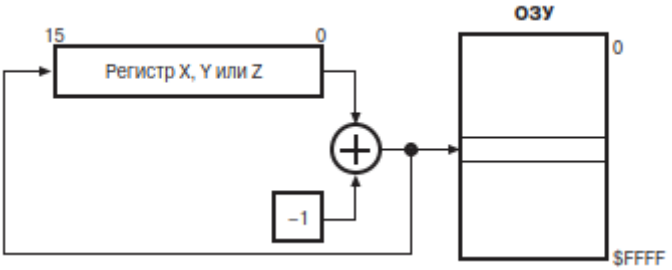


Рис.1.25. Косвенная адресация с преддекрементом

Микроконтроллеры семейства поддерживают 6 команд (по 2 для каждого индексного регистра) косвенной адресации с преддекрементом: LD Rd, –X/–Y/–Z (пересылка байта из ОЗУ в РОН) и ST –X/–Y/–Z, Rd (пересылка байта из РОН в ОЗУ). Адрес регистра общего назначения содержится в битах 8…4 слова команды.

Косвенная адресация с постинкрементом

При использовании команд косвенной адресации с постинкрементом после обращения по адресу, который находится в индексном регистре, содержимое индексного регистра увеличивается на 1 (Рис. 1.26).

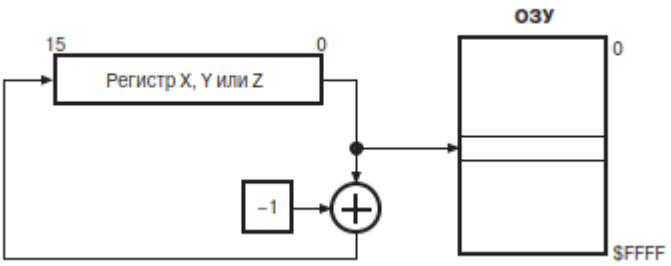


Рис.1.26. Косвенная адресация с постинкрементом

Микроконтроллеры семейства поддерживают 6 команд (по 2 для каждого индексного регистра) косвенной адресации с постинкрементом: LD Rd, X+/Y+/Z+ (пересылка байта из ОЗУ в РОН) и ST X+/Y+/Z+, Rd (пересылка байта из РОН в ОЗУ). Адрес регистра общего назначения содержится в битах 8…4 слова команды.

1.4. Прерывания. Общие сведения

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 PC 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.

Поскольку основными источниками прерываний являются различные периферийные устройства микроконтроллеров, количество прерываний зависит от конкретной модели.

Таблица векторов прерываний

Микроконтроллеры AVR семейства Mega имеют многоуровневую систему приоритетных прерываний. Младшие адреса памяти программ, начи- ная с адреса $0001 (модели ATmega8515x/8535x, ATmega8x и ATmega48x/88x) или $0002 (остальные модели), отведены под таблицу век- торов прерывания. Каждому прерыванию соответствует адрес в этой таб- лице, который загружается в счетчик команд при возникновении преры- вания. Положение вектора в таблице также определяет и приоритет соот- ветствующего прерывания: чем меньше адрес, тем выше приоритет прерывания. Размер вектора прерывания зависит от объема памяти про- грамм микроконтроллера и составляет 1 байт для моделей с объемом памя- ти меньше 16 Кбайт и 2 байта для остальных моделей. Соответственно, для перехода к подпрограммам обработки прерываний в моделях ATmega8515x/8535x, ATmega8x и ATmega48x/88x используются команды RJMP, а в остальных моделях — команды JMP.

Практически во всех микроконтроллерах семейства Mega, за исключением модели ATmega48x, положение таблицы векторов прерываний может быть изменено. Таблица может располагаться не только в начале памяти программ, но также и в начале области загрузчика, причем перемещение таблицы может быть осуществлено непосредственно в ходе выполнения программы.

В зависимости от модели (**Табл. 1.14**) для управления размещением таблицы прерываний используется либо регистр управления микроконтроллера MCUCR, расположенный по адресу $35 ($55), либо общий регистр управления прерываниями GICR, расположенный по адресу $3B ($5B).

Таблица 1.14 **Регистры для управления размещением таблицы прерываний**

Название	Описание	Адрес	Модель
MCUCR	Регистр управления микроконтроллера	$35 ($55)	ATmega64x/128x, ATmega88x/168x, ATmega164x/324x/644x, ATmega165x/325x/3250x/645x/6450x, ATmega640x/1280x/1281x/2560x/2561x
GICR	Общий регистр управления прерываниями	$3B ($5B)	ATmega8515x/8535x, ATmega8x/16x/32x, ATmega162x

Для управления таблицей прерываний в этих регистрах используются два младших бита: IVSEL (1-й бит) и IVCE (0-й бит). Состояние флага IVSEL определяет положение таблицы в памяти программ. Если флаг сбро- шен в 0, то таблица векторов прерываний располагается в начале памяти программ, если установлен в 1 — в начале области загрузчика. Конкретное значение начального адреса области загрузчика зависит от установок конфигурационных ячеек BOOTSZ1 и BOOTSZ0 (кроме моделей ATmega48x). Бит IVCE предназначен для разрешения изменения флага IVSEL.

Для изменения положения таблицы векторов прерываний необходимо выполнить следующие действия:

1. Установить бит IVCE в 1.
2. В течение следующих четырех тактов занести требуемое значение в бит IVSEL, при этом бит IVCE сбрасывается в 0. В противном случае бит IVCE буден сброшен аппаратно по истечении четырех тактов, запрещая дальнейшее изменение флага IVSEL.

На время выполнения описанной последовательности прерывания автоматически запрещаются и разрешаются только после сброса флага IVCE. Состояние флага I регистра SREG при этом не меняется.

Размер таблицы зависит от модели микроконтроллера и составляет от 16 (модели ATmega8515x) до 56 (модели ATmega640x/1280x/2560x) векторов. Распределение адресов таблицы векторов прерываний для различных микроконтроллеров семейства приведено в Табл.1.15. При размещении векторов прерываний в области загрузчика к значениям, указанным в таб- лицах, следует прибавить значение начального адреса области загрузчика.

Таблица 1.15. **Таблица векторов прерываний моделей ATmega48x/88x/168x**

Источник	**Описание**		**Адрес**	
			ATmega48x/88x	**ATmega168x**
INT0	Внешнее прерывание 0		$0001	$0002
INT1	Внешнее прерывание 1		$0002	$0004
PCINT0	Прерывание 0 по изменению состояния выводов		$0003	$0006
PCINT1	Прерывание 1 по изменению состояния выводов		$0004	$0008
PCINT2	Прерывание 2 по изменению состояния выводов		$0005	$000A
WDT	Тайм-аут сторожевого таймера		$0006	$000C

TIMER2 COMPA	Совпадение A таймера/счетчика T2		$0007	$000E
TIMER2 COMPB	Совпадение B таймера/счетчика T2		$0008	$0010
TIMER2 OVF	Переполнение таймера/счетчика T2		$0009	$0012
TIMER1 CAPT	Захват таймера/счетчика T1	0	$000A	$0014
TIMER1 COMPA	Совпадение A таймера/счетчика T1	1	$000B	$0016
TIMER1 COMPB	Совпадение B таймера/счетчика T1	2	$000C	$0018
TIMER1 OVF	Переполнение таймера/счетчика T1	3	$000D	$001A
TIMER0 COMPA	Совпадение A таймера/счетчика T0	4	$000E	$001C
TIMER0 COMPB	Совпадение B таймера/счетчика T0	5	$000F	$001E
TIMER0 OVF	Переполнение таймера/счетчика T0	6	$0010	$0020
SPI, STC	Передача по SPI завершена	7	$0011	$0022
USART, RXC	USART, прием завершен	8	$0012	$0024
USART, UDRE	Регистр данных USART пуст	9	$0013	$0026
USART, TXC	USART, передача завершена	0	$0014	$0028
ADC	Преобразование АЦП завершено	1	$0015	$002A
EE_RDY	EEPROM готово	2	$0016	$002C
ANA_COMP	Аналоговый компаратор	3	$0017	$002E
TWI	Прерывание от модуля TWI	4	$0018	$0030
SPM_RDY	Готовность SPM	5	$0019	$0032

Обработка прерываний

Для глобального разрешения/запрещения прерываний предназначен флаг I регистра SREG. Для разрешения прерываний он должен быть уста- новлен в 1, а для запрещения — сброшен в 0. Индивидуальное разрешение или запрещение (маскирование) прерываний производится установ- кой/сбросом соответствующих битов регистров масок прерываний, рас- сматриваемых ниже.

При возникновении прерывания флаг I регистра SREG аппаратно сбрасывается, запрещая тем самым обработку следующих прерываний. Однако в подпрограмме обработки прерывания этот флаг можно снова ус- тановить в 1 для разрешения вложенных прерываний. При возврате из подпрограммы обработки прерывания (при выполнении команды RETI) флаг I устанавливается аппаратно.

Все имеющиеся прерывания можно разделить на два типа. Прерыва- ния первого типа генерируются при наступлении некоторого события, в результате которого устанавливается флаг прерывания. Затем, если преры- вание разрешено, в счетчик команд загружается адрес вектора соответ- ствующего прерывания. При этом флаг прерывания аппаратно сбрасыва- ется. Он также может быть сброшен программно, записью лог. 1 в бит ре- гистра, соответствующий флагу.

Прерывания второго типа не имеют флагов прерываний и генерируют- ся в течение всего времени, пока присутствуют условия, необходимые для генерации прерывания. Соответственно, если условия, вызывающие пре- рывание, исчезнут до разрешения прерывания, генерации прерывания не произойдет.

Следует помнить, что при вызове подпрограмм обработки прерываний регистр состояния SREG не сохраняется. Поэтому пользователь должен самостоятельно запоминать содержимое этого регистра при входе в под- программу обработки прерывания (если это необходимо) и восстанавли- вать его значение перед вызовом команды RETI.

Микроконтроллеры семейства Mega поддерживают очередь прерыва- ний, которая работает следующим образом: если условия генерации одно- го или более прерываний возникают в то время, когда флаг общего разре- шения прерываний сброшен (все прерывания запрещены), соответствую- щие флаги устанавливаются в 1 и остаются в этом состоянии до установки флага общего разрешения прерываний. После разрешения прерываний выполняется их обработка в порядке приоритета.

Наименьшее время отклика для любого прерывания составляет 5 так- тов в моделях ATmega2560x/2561x и 4 такта — в остальных моделях. В тече- ние этого времени происходит сохранение счетчика команд в стеке. В те- чение

последующих двух (для моделей ATmega8515x/8535x, ATmega8x и ATmega48x/88x) или трех тактов выполняется команда перехода к подпрограмме обработки прерывания. Если прерывание произойдет во время выполнения команды, длящейся несколько циклов, то генерация прерыва- ния произойдет только после выполнения этой команды. Если же преры- вание произойдет во время нахождения микроконтроллера в «спящем» режиме, то время отклика увеличивается еще на 4 или 5 тактов.

Возврат в основную программу занимает 4 такта (в моделях ATmega2560x/2561x — 5 тактов), в течение которых происходит восстановление счетчика команд из стека. После выхода из прерывания процессор всегда выполняет одну команду основной программы, прежде чем обслу-жить любое отложенное прерывание.

Вопросы по 1 главе:

1. Какие основные типы памяти (FLASH, ОЗУ, EEPROM) имеются в микроконтроллерах семейства Mega и каковы их типовые объёмы для моделей ATmega48/88/168?
2. Какие существуют способы адресации памяти данных в AVR? Приведите примеры команд для каждого способа.
3. Что такое таблица векторов прерываний, где она располагается и как определяется приоритет прерываний?
4. Какие регистры образуют 16-битные указатели X, Y, Z и для каких режимов адресации они применяются?
5. Какие команды позволяют работать с регистрами ввода/вывода (IN, OUT, SBI, CBI, SBIC, SBIS)? Какие ограничения на адресацию существуют?
6. Каков порядок подключения внешнего ОЗУ к микроконтроллеру и какие регистры управляют его работой (SRE, XMCRA, XMCRB)?

Глава 2. Порты ввода/вывода.

2.1. Общие сведения

Каждый порт микроконтроллеров состоит из определенного числа выводов, через которые микроконтроллер может осуществлять прием и пере- дачу цифровых сигналов. Задание направления передачи данных через лю- бой контакт ввода/вывода может быть произведено программно в любой момент времени.

Выходные буферы всех портов, имея симметричные нагрузочные характеристики, обеспечивают высокую нагрузочную способность при лю- бом уровне сигнала. Нагрузочной способности достаточно для непосред- ственного управления светодиодными индикаторами.

Входные буферы всех выводов построены по схеме триггера Шмитта. Для всех входов имеется возможность подключения внутреннего подтягивающего резистора между входом и шиной питания VCC.

Отличительной особенностью портов микроконтроллеров семейства Mega (как и всех микроконтроллеров AVR) при использовании их в качес- тве цифровых портов ввода/вывода общего назначения является реализа- ция истинной функциональности «чтение/модификация/запись». Благо- даря этому можно выполнять операции над любым выводом (с помощью команд SBI и CBI), не влияя на другие выводы порта. Это относится к из- менению режима работы контакта ввода/вывода, к изменению состояния выходного буфера (для выходов) и к изменению состояния внутреннего подтягивающего резистора (для входов).

Микроконтроллеры различных моделей семейства имеют разное число портов и соответственно контактов ввода/вывода. Эти данные приведены в Табл. 2.1.

Порт ввода/вывода	ATmega8515x	ATmega8535x	ATmega8x	ATmega16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x	ATmega325x/645x	ATmega3250x/6450x	ATmega1281x/2561x	ATmega640x, ATmega1280x/2560x
PORTA	●	●	—	●	●	—	●	●	●	●	●	●	●
PORTB	●	●	●	●	●	●	●	●	●	●	●	●	●
PORTC	●	●	7	●	●	7	●	●	●	●	●	●	●
PORTD	●	●	●	●	●	●	●	●	●	●	●	●	●
PORTE	3	—	—	—	●	—	3	—	●	●	●	●	●
PORTF	—	—	—	—	●	—	—	—	●	●	●	●	●
PORTG	—	—	—	—	5	—	—	—	5	6[1)]	6[1)]	6	6
PORTH	—	—	—	—	—	—	—	—	—	—	●	—	●
PORTJ	—	—	—	—	—	—	—	—	—	—	7	—	●
PORTK	—	—	—	—	—	—	—	—	—	—	—	—	●
PORTL	—	—	—	—	—	—	—	—	—	—	—	—	●
Число контактов ввода/вывода	35	32	23	32	53	23	35	32	53	53	68	86	51

[1)] Вывод PG5 — только вход с постоянно включенной подтяжкой (совмещен с входом $\overline{\text{RESET}}$).

Примечание. Цифра обозначает разрядность порта (отличную от 8).

Табл.2.1. Порты ввода/вывода микроконтроллеров семейства Mega

Регистры портов ввода/вывода

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 пор- та PORTx, регистр направления данных DDRx и регистр выводов порта PINx. Действительные названия регистров получаются подстановкой на- звания порта вместо символа x. Соответственно, регистры порта A называ- ются 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. Тем не менее, в новых моделях микроконтроллеров (все модели, кроме ATmega8515x/8535x, ATmega8x/16x/32x/64x/128x и ATmega162x) запись 1 в бит регистра PINx приводит к переключению состояния соответствующе- го бита регистра данных PORTx.

Адреса регистров всех портов ввода/вывода приведены в Табл. 2.2

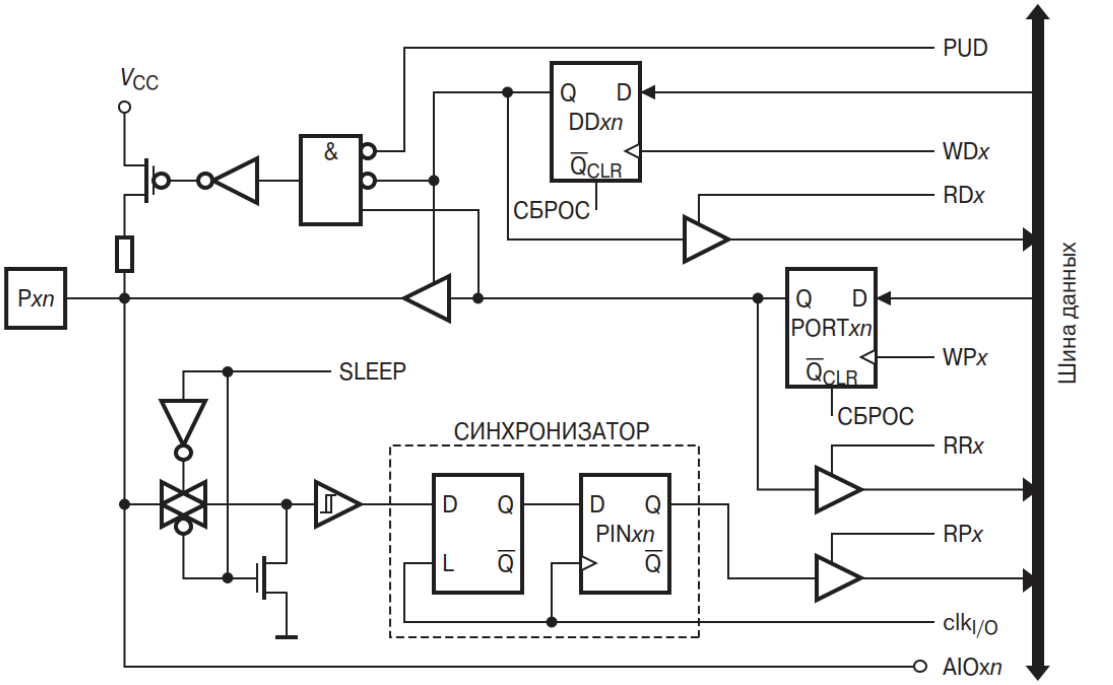
Порт	Регистр	ATmega8515x	ATmega8535x	ATmega8x	ATmega16x	ATmega162x	ATmega64x, ATmega128x	ATmega48x/88x/168x	ATmega164x/324x/644x	ATmega165x, ATmega325x/645x	ATmega3250x/6450x	ATmega1281x/2561x	ATmega640x, ATmega1280x/2560x
A	PORTA	$1B ($3B)		—	$1B ($3B)			—	$02 ($22)				
	DDRA	$1A ($3A)		—	$1A ($3A)			—	$01 ($21)				
	PINA	$19 ($39)		—	$19 ($39)			—	$00 ($20)				
B	PORTB	$18 ($38)						$05 ($25)					
	DDRB	$17 ($37)						$04 ($24)					
	PINB	$16 ($36)						$03 ($23)					
C	PORTC	$15 ($35)						$08 ($28)					
	DDRC	$14 ($34)						$07 ($27)					
	PINC	$13 ($33)						$06 ($26)					
D	PORTD	$12 ($32)						$0B ($2B)					
	DDRD	$11 ($31)						$0A ($2A)					
	PIND	$10 ($30)						$09 ($29)					
E	PORTE	$07 ($27)		—	—	$07 ($27)	$03 ($23)	—	—	$0E ($2E)			
	DDRE	$06 ($26)		—	—	$06 ($26)	$02 ($22)	—	—	$0D ($2D)			
	PINE	$05 ($25)		—	—	$05 ($25)	$01 ($21)	—	—	$0C ($2C)			

Табл.2.2. Регистры портов ввода/вывода

Порт	Регистр	ATmega8515x	ATmega8535x	ATmega8x	ATmega16x	ATmega162x	ATmega64x, ATmega128x	ATmega48x/88x/168x	ATmega164x/324x/644x	ATmega165x, ATmega325x/645x	ATmega3250x/6450x	ATmega1281x/2561x	ATmega640x, ATmega1280x/2560x
F	PORTF	$03 ($23)	—	—	—	—	($62)	—	—	$11 ($31)			
	DDRF	$02 ($22)	—	—	—	—	($61)	—	—	$10 ($30)			
	PINF	$01 ($21)	—	—	—	—	$00 ($20)	—	—	$0F ($2F)			
G	PORTG	—	—	—	—	—	($65)	—	—	$14 ($34)			
	DDRG	—	—	—	—	—	($64)	—	—	$13 ($33)			
	PING	—	—	—	—	—	($63)	—	—	$12 ($32)			
H	PORTH	—	—	—	—	—	—	—	—	—	($DA)	—	($102)
	DDRH	—	—	—	—	—	—	—	—	—	($D9)	—	($101)
	PINH	—	—	—	—	—	—	—	—	—	($D8)	—	($100)
J	PORTJ	—	—	—	—	—	—	—	—	—	($DD)	—	($105)
	DDRJ	—	—	—	—	—	—	—	—	—	($DC)	—	($104)
	PINJ	—	—	—	—	—	—	—	—	—	($DB)	—	($103)
K	PORTK	—	—	—	—	—	—	—	—	—	—	—	($108)
	DDRK	—	—	—	—	—	—	—	—	—	—	—	($107)
	PINK	—	—	—	—	—	—	—	—	—	—	—	($106)
L	PORTL	—	—	—	—	—	—	—	—	—	—	—	($10B)
	DDRL	—	—	—	—	—	—	—	—	—	—	—	($10A)
	PINL	—	—	—	—	—	—	—	—	—	—	—	($109)

Конфигурирование портов ввода/вывода

Упрощенная структурная схема одного из каналов порта ввода/вывода Pxn при работе его в качестве цифрового входа/выхода общего назначения приведена на Рис. 2.1.



PUD — выключение подт. резистора
SLEEP — управление спящим режимом
$clk_{I/O}$ — тактовый сигнал подсистемы ввода/вывода

WD*x* — запись регистра DDR*x*
RD*x* — чтение регистра DDR*x*
WP*x* — запись регистра PORT*x*
RR*x* — чтение регистра PORT*x*
RP*x* — чтение выводов порта PORT*x*

AIO*xn* — аналоговый вход/выход вывода *n* порта PORT*x*

Примечание. Сигналы WP*x*, WD*x*, RP*x*, RD*x* являются общими для всех выводов одного порта; сигналы $clk_{I/O}$, SLEEP и PUD являются общими для всех портов микроконтроллера.

Рис.2.1. Структурная схема канала ввода/вывода

Каждому выводу порта соответствуют три бита регистров ввода/выво- да: PORTxn (регистр PORTx), DDxn (регистр DDRx) и PINxn (регистр PINx). Действительные названия битов регистров получаются подстанов- кой названия порта вместо символа x и номера бита вместо символа n. По- рядковый номер вывода порта соответствует порядковому номеру бита ре- гистров этого порта. Поэтому, если разрядность порта меньше восьми, в регистрах порта используется соответствующее число младших битов. Не- задействованные старшие биты регистров доступны только для чтения и всегда содержат 0.

Бит DDxn регистра DDx определяет направление передачи данных че- рез контакт ввода/вывода. Если этот бит установлен в 1, то n-й вывод пор- та является выходом, если же сброшен в 0 — входом.

Бит PORTxn регистра PORTx выполняет двойную функцию. Если вы- вод функционирует как выход (DDxn = 1), то этот бит определяет состоя- ние вывода порта. Если бит установлен в 1, на выводе устанавливается на- пряжение ВЫСОКОГО уровня. Если бит сброшен в 0, на выводе устанав- ливается напряжение НИЗКОГО уровня.

Если же вывод функционирует как вход (DDxn = 0), то бит PORTxn оп- ределяет состояние внутреннего подтягивающего резистора для данного

вывода. При установке бита PORTxn в 1 подтягивающий резистор подключается между выводом микроконтроллера и линией питания.

Вообще говоря, управление подтягивающими резисторами во всех микроконтроллерах семейства осуществляется на двух уровнях. Общее управление (для всех выводов портов) осуществляется битом PUD регистра специальных функций SFIOR или регистра управления микроконтролле- ра MCUCR (в зависимости от модели). В моделях ATmega64x и ATmega128x регистр SFIOR располагается по адресу $20 ($40), а в осталь- ных моделях — по адресу $30 ($50). Регистр MCUCR располагается по ад- ресу $35 ($55). Форматы этих регистров приведены на Рис. 2.2.

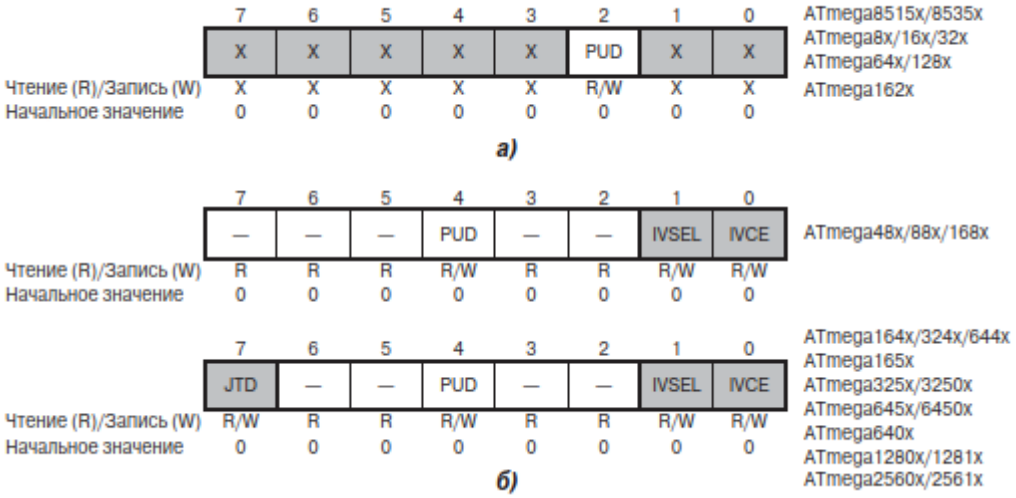


Рис.2.2. Регистры управления подтяжкой SFIOR (а) и MCUCR (б)

Если бит PUD сброшен в 0 (начальное состояние), то состояние подтягивающих резисторов будет определяться состоянием битов PORTxn для каждого входа порта. Если же бит PUD установлен в 1, подтягивающие резисторы отключаются от всех выводов микроконтроллера.

Обратите внимание, что при переключении вывода между третьим состоянием (DDxn = 0, PORTxn = 0) и состоянием ВЫСОКОГО уровня (DDxn = 1, PORTxn = 1) происходит переход через одно из промежуточ- ных состояний: либо включается подтягивающий резистор (DDxn = 0, PORTxn = 1), либо выход переключается в состояние НИЗКОГО уровня (DDxn = 1, PORTxn = 0). Наиболее применимым является, как правило, первый вариант, поскольку для высокоимпедансных систем безразлично, каким образом формируется ВЫСОКИЙ уровень. Если в каком-либо слу чае это не подходит, пользователь может отключить подтягивающие резис- торы от всех портов установкой бита PUD в 1.

Аналогичная ситуация возникает и при переключении между состоянием с включенным подтягивающим резистором (DDxn = 0, PORTxn = 1) и состоянием НИЗКОГО уровня (DDxn = 1, PORTxn = 0). В этом случае промежуточным состоянием является либо высокоимпедансное со- стояние

(DDxn = 0, PORTxn = 0), либо состояние ВЫСОКОГО уровня (DDxn = 1, PORTxn = 1).

Все возможные сочетания состояний управляющих битов и соответственно конфигурации выводов портов приведены в Табл. 2.3

Таблица 2.3. **Конфигурации выводов портов**

DD*xn*	**PORT *xn***	**PUD**	**Функция вывода**	**Резистор**	**Примечание**
0	0	X	Вход	Отключен	Третье состояние (Hi-Z)[1)]
0	1	0	Вход	Подключен	При подключении нагрузки между выво- дом и общим проводом вывод является ис- точником тока
0	1	1	Вход	Отключен	Третье состояние (Hi-Z)
1	0	X	Выход	Отключен	Выход установлен в 0
1	1	X	Выход	Отключен	Выход установлен в 1

[1)] Состояние выводов портов при сбросе.

Состояние вывода микроконтроллера (независимо от установок бита DDxn) может быть получено путем чтения бита PINxn регистра PINx. При этом следует помнить, что между действительным изменением сигнала на выводе и изменением бита PINxn существует задержка. Эта задержка вно- сится узлом синхронизации, состоящим, как показано на Рис. 2.1, из бита PINxn и дополнительного триггера-защелки. Значение сигнала на выводе микроконтроллера фиксируется триггером-защелкой при НИЗКОМ уров- не тактового сигнала и переписывается затем в бит PINxn по нарастающе- му фронту тактового сигнала. Соответственно, величина задержки может составлять от 0.5 до 1.5 периодов системного тактового сигнала, как пока- зано на Рис. 2.3, а.

По этой же причине между операциями изменения и повторного считывания состояний вывода необходимо вставлять команду NOP. Поскольку

команда OUT устанавливает сигнал «SYNC LATCH» в 1 по положительно- му фронту тактового сигнала, задержка в этом случае равна одному перио- ду тактового сигнала (Рис. 2.3, б).

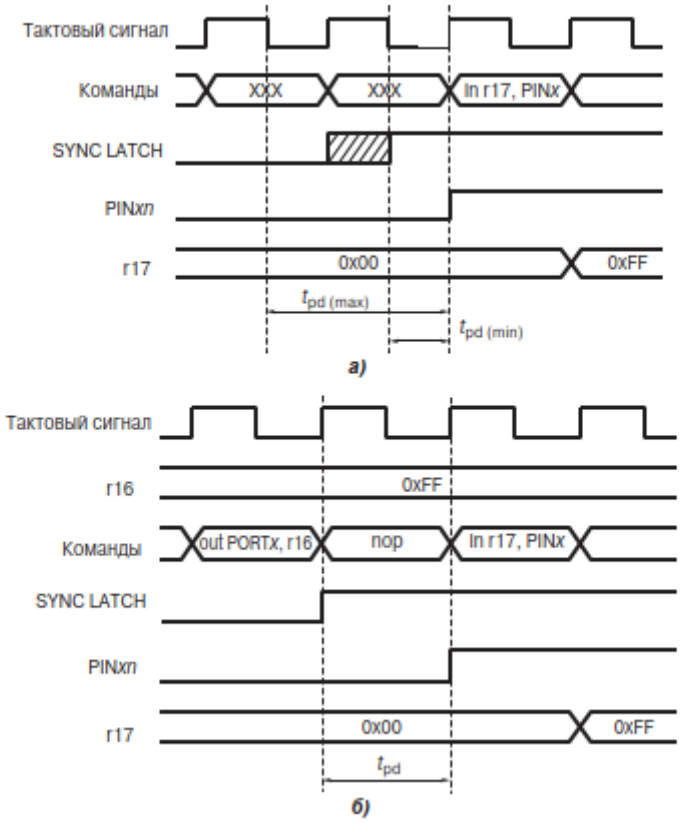


Рис.2.3. Синхронизация при чтении состояния вывода:
а — при считывании бита PINxn; б — при считывании со- стояния вывода, заданного программнно

Далее приведен пример конфигурирования одного из портов микро- контроллера. В примере выводы 0 и 1 порта B устанавливаются в 1, выводы 2 и 3 — в 0. Выводы 4…7 порта конфигурируются как входы, при этом к вы- водам 6 и 7 подключаются подтягивающие резисторы.

Пример на ассемблере

```
...

ldi r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)

ldi r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)

out PORTB,r16    ; Задать состояние выходов и подтягиваюших

; резисторов

out DDRB,r17     ; Задать режимы работы выводов

nop  ; для синхронизации
```

```
in     r16,PINB     ; Считать состояние выводов порта
```

Пример на Си

```
unsigned char i;

...

/* Задать состояние выходов и подтягивающих резисторов */

/* Задать режимы работы выводов */

PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0); DDRB =
(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);

_NOP();     /* Синхронизация */

i = PINB;   /* Считать состояние выводов порта */

...
```

В заключение отметим, что подавляющее большинство контактов ввода/вывода всех микроконтроллеров семейства имеют дополнительные функции и могут использоваться различными периферийными устрой- ствами микроконтроллеров. При этом возможны две ситуации. В одних случаях пользователь должен самостоятельно задавать конфигурацию вы- вода, а в других вывод конфигурируется автоматически при включении соответствующего периферийного устройства. Об этом будет сказано при рассмотрении соответствующих периферийных устройств.

2.2. Таймеры

Микроконтроллеры семейства в зависимости от модели имеют в своем составе от двух до шести таймеров/счетчиков общего назначения (Табл. 2.4).

Таблица 2.4. **Таймеры/счетчики общего назначения**

Таймер/счетчик	ATmega8515x	ATmega8535x	ATmega8x, ATmega16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x	ATmega325x/3250x, ATmega645x/6450x	ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x

Таймер/счетчик T0 (8-битный)				•)	•	•	•	•	•	•
Таймер/счетчик T1 (16-битный)	•	•	•	•	•	•	•	•	•	•
Таймер/счетчик T2 (8-битный)		•)	•)	•	•)	•)	•)	•)	•)	•)
Таймер/счетчик T3 (16-битный)				•		•				•
Таймер/счетчик T4 (16-битный)										•
Таймер/счетчик T5 (16-битный)										•
[1)] Асинхронный таймер/счетчик.										

Как видно из таблицы, во всех моделях микроконтроллеров семейства присутствуют как минимум два таймера/счетчика — T0 и T1. Таймер/счет- чик T0 имеет минимальный набор функций, зависящий, тем не менее, от модели микроконтроллера. В одних моделях он может использоваться только для отсчета и измерения временных интервалов или как счетчик внешних событий. В других моделях к этим функциям добавляется воз- можность генерации сигналов с широтно-импульсной модуляцией (ШИМ) фиксированной разрядности (один или два канала), а также воз- можность работать в асинхронном режиме в качестве часов реального вре- мени (в моделях ATmega64x/128x).

Таймер/счетчик T1 тоже может использоваться для отсчета временных интервалов и как счетчик внешних событий. Кроме того, он может осу- ществлять запоминание своего состояния по внешнему сигналу. Как и таймер/счетчик T0, он может работать в качестве 2- или 3-канального ши- ротно-импульсного модулятора, но уже переменной разрядности. Коли- чество каналов ШИМ зависит от модели.

Таймер/счетчик T2 практически полностью аналогичен таймеру/счет- чику T0. Во всех моделях, кроме ATmega64x/128x, таймер/счетчик T2 мо- жет работать в асинхронном режиме.

Таймеры/счетчики T3…T5 по функциональным возможностям иден- тичны таймеру/счетчику T1.

В составе всех микроконтроллеров семейства имеется также сторо- жевой таймер, являющийся непременным атрибутом всех современных микроконтроллеров. Этот таймер позволяет избежать несанкциониро- ванного зацикливания программы, возникающего по тем или иным при- чинам.

Назначение выводов таймеров/счетчиков

Каждый таймер/счетчик использует один или более выводов микро- контроллера. Как правило, эти выводы — линии портов ввода/вывода об- щего назначения, а функции, реализуемые этими выводами при работе совместно с таймерами/счетчиками, являются их альтернативными функ- циями.

Все выводы микроконтроллеров, используемые таймерами/счетчика- ми

общего назначения, приведены в Табл. 2.5. Там же указаны функции этих выводов.

Не забывайте о том, что при использовании альтернативных функций линий портов ввода/вывода необходимо, как правило, самостоятельно сконфигурировать эти выводы в соответствии с их функциональным назначением.

Название	ATmega8515x	ATmega8535x	ATmega8x	ATmega16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x/325x/645x	ATmega3250x/6450x	ATmega1281x/2561x	ATmega640x/1280x/2560x	Описание
T0	PB0	PB0	PD4	PB0	—	PD4	PB0	PB0	PG4	PG4	PD7	PD7	Вход внешнего сигнала таймера T0
OC0	PB0	PB3	—	PB3	PB4	—	PB0	—	—	—	—	—	Выход схемы сравнения таймера T0
OC0A	—	—	—	—	—	PD6	—	PB3	PB4	PB4	PB7	PB7	
OC0B	—	—	—	—	—	PD5	—	PB4	—	—	PG5	PG5	
T1	PB1	PB1	PD5	PB1	PD6	PD5	PB1	PB1	PG3	PG3	PD6	PD6	Вход внешнего сигнала таймера T1
ICP	PE0	—	—	—	—	—	—	—	—	—	—	—	Вход захвата таймера T1
ICP1	—	PD6	PB0	PD6	PD4	PB0	PE0	PD6	PD0	PD0	PD4	PD4	
OC1A	PD5	PD5	PB1	PD5	PB5	PB1	PD5	PD5	PB5	PB5	PB5	PB5	Выход схемы сравнения таймера T1
OC1B	PE2	PD4	PB2	PD4	PB6	PB2	PE2	PD4	PB6	PB6	PB6	PB6	
OC1C	—		—	—	PB7	—	—	—	—	—	PB7	PB7	
T2	—		—	—	PD7	—	—	—	—	—	—	—	Вход внешнего сигнала таймера T2
OC2	—	—	PB3	PD7	PB7	—	PB1	—	—	—	—	—	Выход схемы сравнения таймера T2
OC2A	—	—	—	—	—	PB3	—	PD7	PB7	PB7	PB4	PB4	
OC2B	—	—	—	—	—	PD3	—	PD6	—	—	—	PH6	
T3	—	—	—	—	PE6	—	—	—	—	—	PE6	PE6	Вход внешнего сигнала таймера T3
ICP3	—	—	—	—	PE7	—	PD3	—	—	—	PE7	PE7	Вход захвата таймера T3

Табл.2.5. Выводы, используемые таймерами/счетчиками общего назначения

(продолжение)

Название	ATmega8515x	ATmega8535x	ATmega8x	ATmega16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x/325x/645x	ATmega3250x/6450x	ATmega1281x/2561x	ATmega640x/1280x/2560x	Описание
OC3A	—	—	—	—	PE3	—	PD4	—	—	—	PE3	PE3	Выход схемы сравнения таймера T3
OC3B	—	—	—	—	PE4	—	PB4	—	—	—	PE4	PE4	
OC3C	—	—	—	—	PE5	—	—	—	—	—	PE5	PE5	
T4	—	—	—	—	—	—	—	—	—	—	—	PH7	Вход внешнего сигнала таймера T4
ICP4	—	—	—	—	—	—	—	—	—	—	—	PL0	Вход захвата таймера T4
OC4A	—	—	—	—	—	—	—	—	—	—	—	PH3	Выход схемы сравнения таймера T4
OC4B	—	—	—	—	—	—	—	—	—	—	—	PH4	
OC4C	—	—	—	—	—	—	—	—	—	—	—	PH5	
T5	—	—	—	—	—	—	—	—	—	—	—	PL2	Вход внешнего сигнала таймера T5
ICP5	—	—	—	—	—	—	—	—	—	—	—	PL1	Вход захвата таймера T5
OC5A	—	—	—	—	—	—	—	—	—	—	—	PL3	Выход схемы сравнения таймера T5
OC5B	—	—	—	—	—	—	—	—	—	—	—	PL4	
OC5C	—	—	—	—	—	—	—	—	—	—	—	PL5	
TOSC1	—	PC6	PB6	PC6	PG4	PB6	PD4	PC6	—[1)]	—[1)]	PG4	PG4	Вход для подключения резонатора
TOSC2	—	PC7	PB7	PC7	PG3	PB7	PD5	PC7	—[1)]	—[1)]	PG3	PG3	Выход для подключения резонатора

[1)] Отдельные контакты ввода/вывода, совмещенные с выводами XTAL1/XTAL2.

2.3. Прерывания от таймеров/счетчиков

В старых моделях для разрешения/запрещения прерываний от таймеров/счетчиков использовалось от одного до двух регистров ввода/вывода. В новых моделях число таких регистров (3…6) равно числу счетчиков в конкретной модели. Точно так же дело обстоит и с регистрами, содержащими флаги прерываний. Названия и адреса всех этих регистров приведены в **Табл. 2.6**.

Таблица 2.6. **Регистры для управления прерываниями от таймеров/счетчиков**

Модель	**Таймер/счетчик**	**Разрешение прерываний**		**Флаги прерываний**	
		Регистр	**Адрес**	**Регистр**	**Адрес**
ATmega8515x	T0, T1	TIMSK	$39 ($59)	TIFR	$38 ($58)
ATmega8535x	T0, T1, T2	TIMSK	$39 ($59)	TIFR	$38 ($58)
ATmega8x/16x/32x	T0, T1, T2	TIMSK	$39 ($59)	TIFR	$38 ($58)
ATmega64x/128x	T0, T1, T2	TIMSK	$37 ($57)	TIFR	$36 ($56)
	T1,T3	ETIMSK	($7D)	ETIFR	($7C)
ATmega162x	T0, T1, T2	TIMSK	$39 ($59)	TIFR	$38 ($58)
	T1,T3	ETIMSK	($7D)	ETIFR	($7C)
ATmega48x/88x/168x, ATmega164x/324x/644x, ATmega165x/325x/645x, ATmega3250x/6450x	T0	TIMSK0	($6E)	TIFR0	$15 ($35)
	T1	TIMSK1	($6F)	TIFR1	$16 ($36)
	T2	TIMSK2	($70)	TIFR2	$17 ($37)
ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x	T0	TIMSK0	($6E)	TIFR0	$15 ($35)
	T1	TIMSK1	($6F)	TIFR1	$16 ($36)
	T2	TIMSK2	($70)	TIFR2	$17 ($37)
	T3	TIMSK3	($71)	TIFR3	$18 ($38)
	T4	TIMSK4	($72)	TIFR4	$19 ($39)
	T5	TIMSK5	($73)	TIFR5	$1A ($3A)

Форматы регистров, используемых для разрешения/запрещения прерываний от таймеров/счетчиков, показаны на **Рис. 2.4…2.**, а описание их битов приведено в **Табл. 2.7**.

Для разрешения какого-либо прерывания от таймера/счетчика

необ- ходимо установить в 1 соответствующий бит регистра TIMSK (TIMSK*n*)/ETIMSK и, разумеется, флаг I регистра SREG.

Таблица 2.7. Биты регистров TIMSK, ETIMSK и TIMSK0…TIMSK5

Название бита	Описание
TOIE*n*	Флаг разрешения прерывания по переполнению таймера/счетчика T*n* (*n* = 0…5)
OCIE*n*	Флаг разрешения прерывания по событию «Совпадение» таймера/счетчика T*n* (*n* = 0, 2)
OCIE*n*A	Флаг разрешения прерывания по событию «Совпадение A» таймера/счетчика T*n* (*n* = 0…5)
OCIE*n*B	Флаг разрешения прерывания по событию «Совпадение B» таймера/счетчика T*n* (*n* = 0…5)
OCIE*n*C	Флаг разрешения прерывания по событию «Совпадение C» таймера/счетчика T*n* (*n* = 1, 3…5)
TICIEI*n*	Флаг разрешения прерывания по событию «Захват» таймера/счетчика T*n* (*n* = 1, 3)
ICIE*n*	Флаг разрешения прерывания по событию «Захват» таймера/счетчика T*n* (*n* = 1, 3…5)

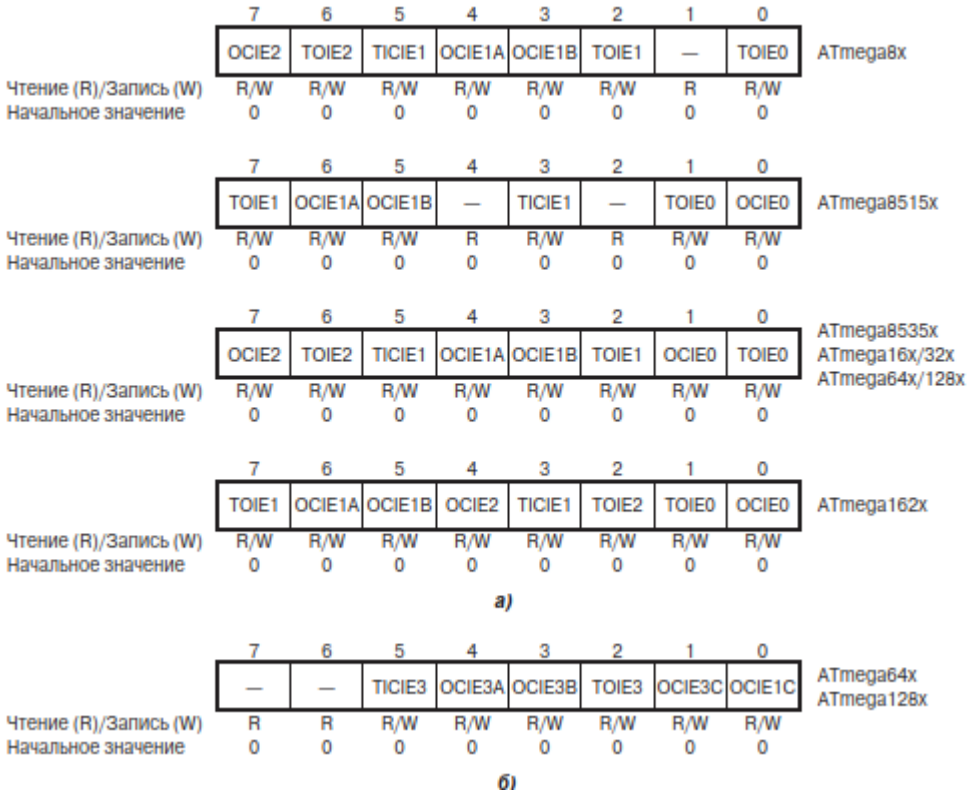


Рис.2.4. Формат регистров TIMSK (а) и ETIMSK (б)

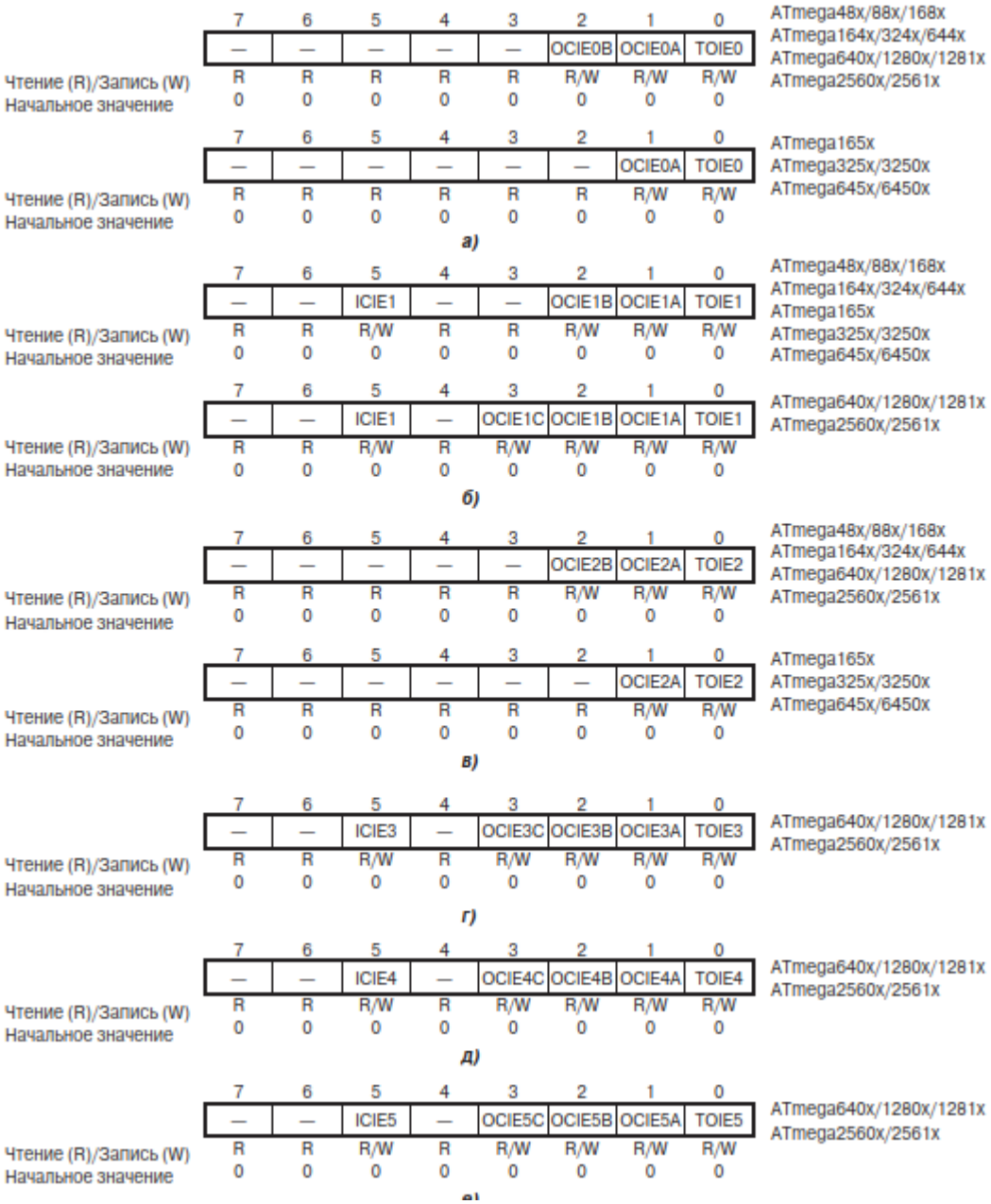


Рис.2.5. Формат регистров TIMSK0 (а), TIMSK1 (б), TIMSK2 (в), TIMSK3 (г), TIMSK4 (д), TIMSK5 (е)

Форматы регистров, используемых для индикации наступления прерываний от таймеров/счетчиков, показаны на Рис. 2.6, а описание их битов приведено в Табл. 2.8.

Таблица 2.8. **Биты регистра TIFR**

Название бита	Описание
TOV*n*	Флаг прерывания по переполнению таймера /счетчика T*n* (*n* = 0…5)

OCF*n*	Флаг прерывания по событию «Совпадение» таймера/счетчика T*n* (*n* = 0, 2)
OCF*n*A	Флаг прерывания по событию «Совпадение A» таймера/счетчика T*n* (*n* = 0…5)
OCF*n*B	Флаг прерывания по событию «Совпадение B» таймера/счетчика T*n* (*n* = 0…5)
OCF*n*C	Флаг прерывания по событию «Совпадение C» таймера/счетчика T*n* (*n* = 1, 3…5)
ICF*n*	Флаг прерывания по событию «Захват» таймера/счетчика T*n* (*n* = 1, 3…5)

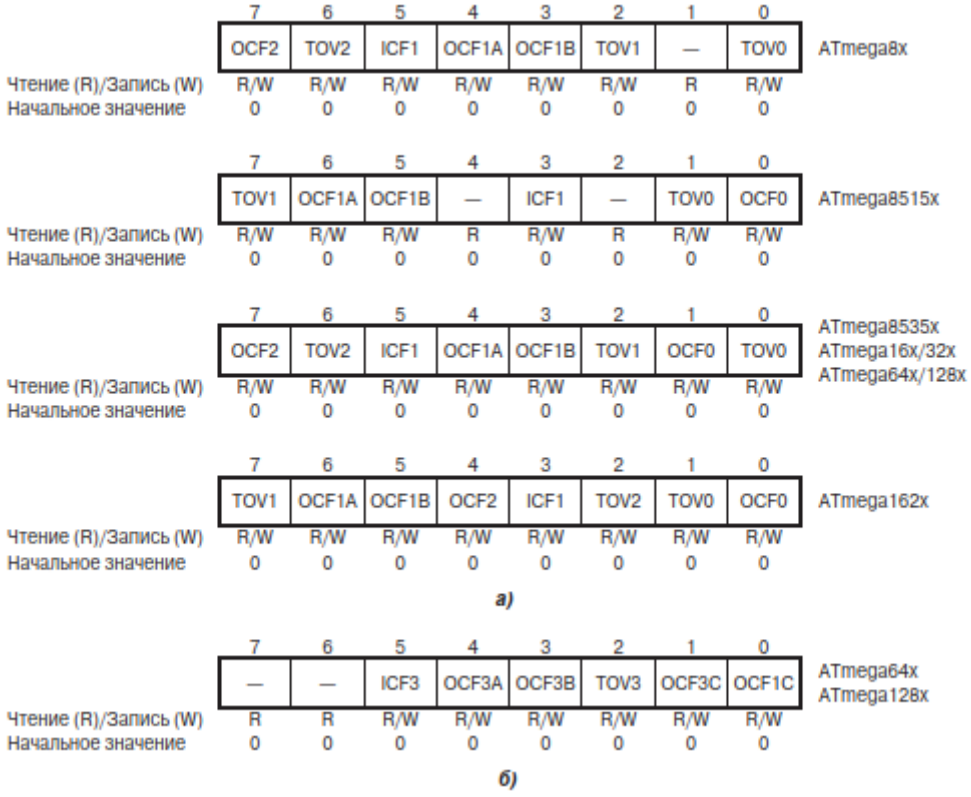


Рис.2.6. Формат регистров TIFR (а) и ETIFR (б)

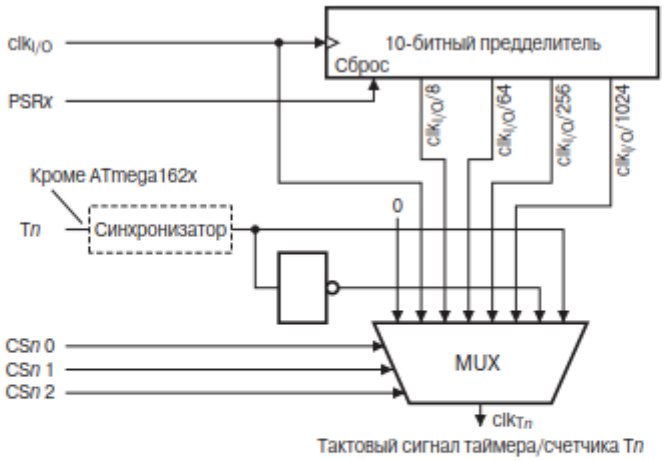
2.4. Предделители таймеров/счетчиков

Блоки предделителей предназначены для формирования тактовых сигналов таймеров/счетчиков clk_{T0}, clk_{T1}, clk_{T2}, clk_{T3}. Упрощенная структурная схема блока предделителя таймеров/счетчиков, не имеющих асинхронного режима работы, приведена на Рис. 2.7, а. Структурная схема блока

преддели- теля таймеров/счетчиков, имеющих возможность работы в асинхронном ре- жиме, приведена на Рис. 2.7, б.

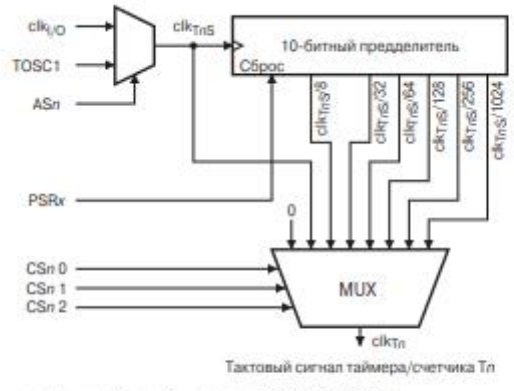
Как показано на рисунке, в состав каждого блока входят собствен- но 10-битный предделитель, выходной мультиплексор (селектор так- тового сигнала), а для таймеров, имеющих возможность работы в асинхронном режиме, — еще и входной мультиплексор исходного так- тового сигнала. По последней схеме выполнен предделитель тайме- ра/счетчика T0 моделей ATmega64x/128x и таймера/счетчика T2 ос- тальных моделей.

Следует иметь в виду, что все таймеры/счетчики каждой модели се- мейства, не имеющие асинхронного режима работы, используют один и тот же 10-битный предделитель. При этом управление тактовым сигналом каждого таймера/счетчика осуществляется индивидуально и будет описа- но при их рассмотрении.



n = 0,1,3 — для ATmega162x (PSRx = PSR310);
1,2,3 — для ATmega64x и ATmega128x (PSRx = PSR321);
0,1 — для ATmega8515x/8535x, ATmega8x/16x/32x и ATmega165x/325x/3250x/645x/6450x (PSRx = PSR10);
0,1 — для ATmega48x/88x/168x и ATmega164x/324x/644x (PSRx = PSRSYNC);
0,1,3,4,5 — для ATmega640x/1280x/1281x/2560x/2561x (PSRx = PSRSYNC).

а)



n = 1 — для ATmega64x и ATmega128x (PSRx = PSR0);
2 — для ATmega8535x, ATmega8x/16x/32x, ATmega162x и ATmega165x/325x/3250x/645x/6450x (PSRx = PSR2);
2 — для ATmega48x/88x/168x, ATmega164x/324x/644x и ATmega640x/1280x/1281x/2560x/2561x (PSRx = PSRASY).

б)

Рис.2.7. Блок предделителя таймеров/счетчиков:
а — без асинхронного режима; б — с асинхронным режимом

Следует понимать, что предделители работают независимо от таймеров/счетчиков. Следствием этого является, в частности, неопределенный промежуток времени (1…N+1 тактов исходного сигнала, где N — коэффициент деления предделителя) между разрешением таймера/счетчика и первым его отсчетом при работе совместно с предделителем. Чтобы уйти от этой неопределенности, можно воспользоваться средствами, описанными в следующем подразделе.

Управление предделителями

Помимо управления тактовым сигналом таймера/счетчика, все микроконтроллеры семейства позволяют осуществлять сброс предделите- лей, а отдельные модели позволяют также осуществлять их остановку. Для этого используется либо регистр специальных функций SFIOR, ли- бо (в новых моделях) регистр управления таймеров/счетчиков GTCCR, расположенный по адресу $23 ($43). Формат этого регистра для различных моделей микроконтроллеров приведен на Рис.2.8. (биты, не используемые для управления предделителями таймеров/счетчиков, указаны на рисунке как X).

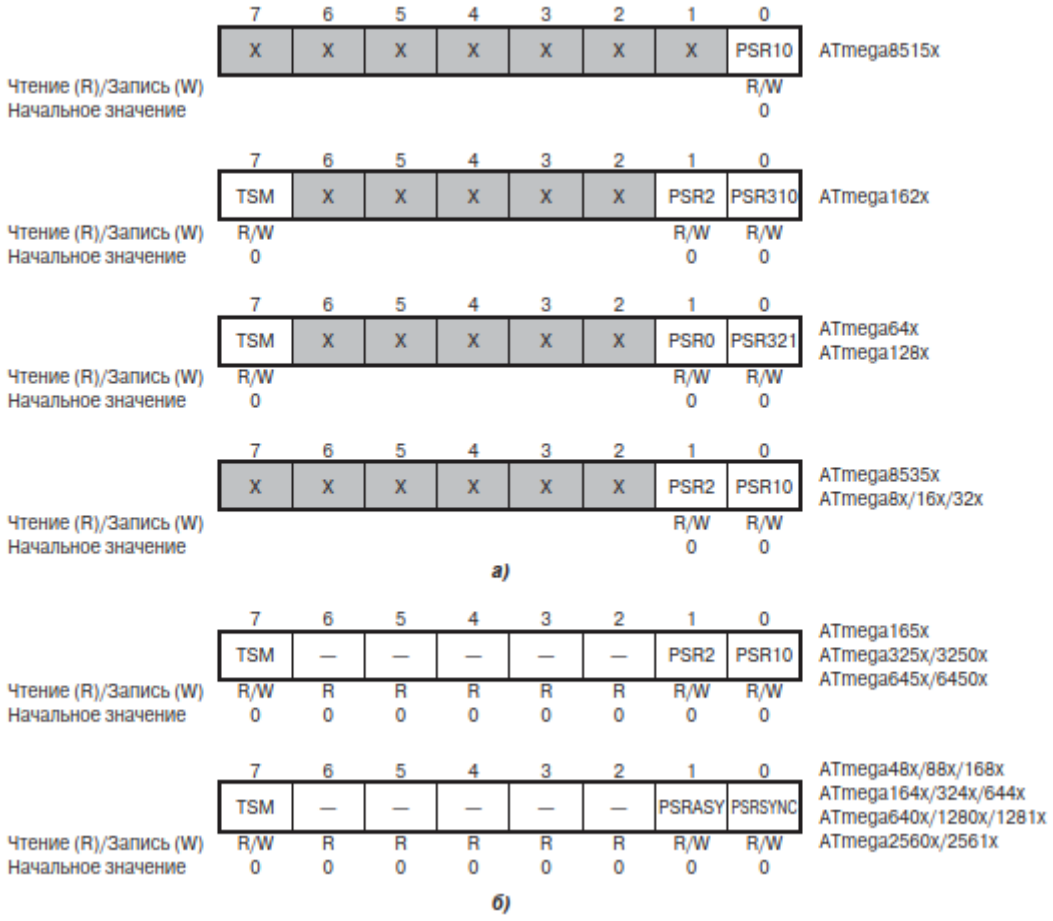


Рис.2.8. Управление предделителями таймеров/счетчиков — регистры SFIOR (а) и GTCCR (б)

Для сброса предделителей таймеров/счетчиков используются биты PSRx (PSRSYNC/PSRASY) регистра. При записи в эти биты лог. 1 предделители соответствующих таймеров/счетчиков переводятся в исходное состояние. Биты

сбрасываются в 0 аппаратно после выполне- ния операции сброса. Напоминаю, что один предделитель, как правило, используется несколькими таймерами/счетчиками, и соответственно сброс предделителя повлияет на все таймеры/счетчики, которые его используют.

Остановка всех предделителей микроконтроллера осуществляется записью лог. 1 в бит TSM регистра SFIOR или GTCCR. Последующий запуск предделителей осуществляется записью в бит TSM лог. 0. Указанная функция может использоваться, в частности, для синхронизации тай меров/счетчиков. После установки бита TSM и битов PSRx (PSRSYNC/PSRASY) соответствующие таймеры/счетчики останавлива- ются и могут быть проинициализированы требуемыми значениями. После сброса бита TSM биты PSRx (PSRSYNC/PSRASY) аппаратно сбрасывают- ся и все таймеры/счетчики начинают работать одновременно.

Использование внешнего тактового сигнала

Практически все таймеры/счетчики, не имеющие асинхронного режима работы, могут тактироваться от внешнего сигнала. Исключение состав- ляет лишь таймер/счетчик T3 модели ATmega162x.

Внешний сигнал, поступающий на вход Tn (n = 0…5) микроконтроллера, прежде чем поступить на вход селектора тактового сигнала, проходит через специальный узел, включающий схему синхронизации и детектор фронтов. В общем виде схема этого узла приведена на Рис. 2.9.

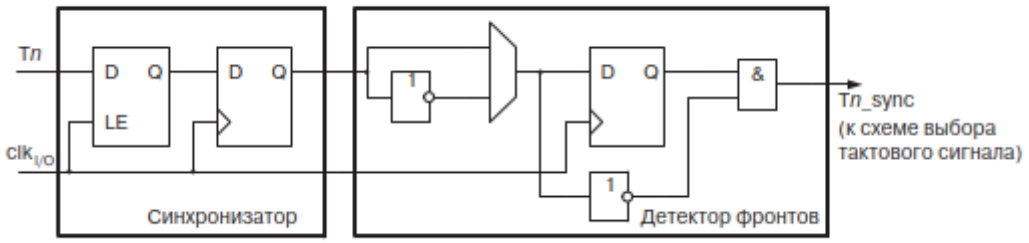


Рис.2.9. Входной каскад выводов Tn

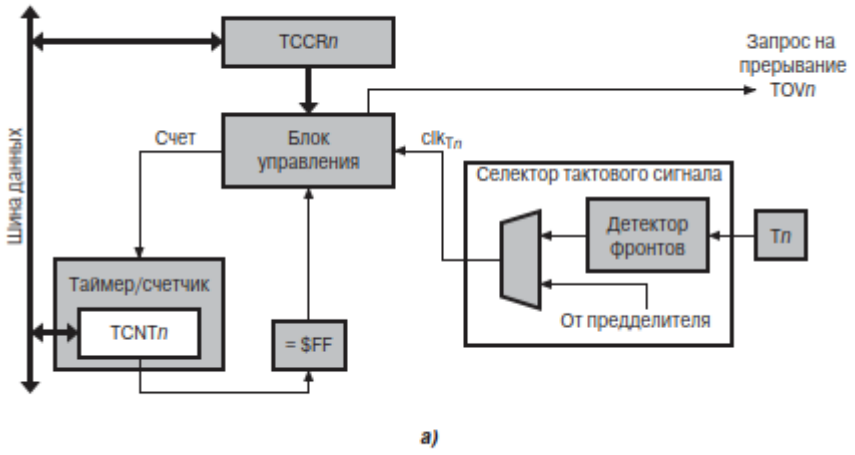
Синхронизация внешнего сигнала осуществляется с частотой так- тового сигнала микроконтроллера (состояние вывода T1 считывается по нарастающему фронту тактового сигнала clkI/O). Поэтому частота внешнего сигнала должна быть в 2 раза ниже частоты тактового сигна- ла микроконтроллера (fEXT < fCLK_I/O/2). Однако, чтобы гарантировать обнаружение фронтов внешнего сигнала во всем диапазоне возмож- ных изменений частоты и скважности тактового сигнала микроконт- роллера (из-за разброса параметров элементов тактового генератора), рекомендуется, чтобы частота внешнего сигнала была меньше, чем fCLK_I/O/2.5.

Также следует понимать, что из-за входного каскада происходит вре-

менная задержка между изменением состояния вывода и обновлением счетного регистра таймера/счетчика. Величина задержки составляет от 2.5 до 3.5 тактов.

2.5. Восьмибитные таймеры/счетчики

Восьмибитный таймер/счетчик T0 присутствует во всех моделях микроконтроллеров семейства Mega, а таймер/счетчик T2 — во всех, кроме ATmega8515x. Всего в микроконтроллерах семейства реализовано пять исполнений восьмибитных таймеров/счетчиков, отличающихся набором выполняемых функций. Самым простым является тай- мер/счетчик T0 в модели ATmega8x (Рис. 2.10, а). Он может использо- ваться только для отсчета временных интервалов или как счетчик внешних событий. Более совершенным является таймер/счетчик T0 моделей ATmega8515x/8535x, ATmega16x/32x, ATmega162x, ATmega165x, ATmega325x/3250x/645x/6450x и таймер/счетчик T2 моде- лей ATmega64x/128x (Рис. 2.10, б). Помимо уже упомянутых функций, эти таймеры/счетчики могут использоваться в качестве одноканального ге- нератора 8-битного ШИМ-сигнала. А в таймере/счетчике T0 остальных моделей (за исключением ATmega64x/128x) имеется уже два независи- мых блока сравнения, что позволяет реализовать 2-канальный генера- тор 8-битного ШИМ-сигнала (Рис. 2.10, в). Несколько особняком стоят таймер/счетчик T0 моделей ATmega64x/128x и таймер/счетчик T2 всех остальных моделей. Основное их отличие заключается в том, что они могут работать в асинхронном режиме (обычно этот режим используется для реализации часов реального времени). Все эти данные сведены в Табл. 2.9 и Табл. 2.10.



а)

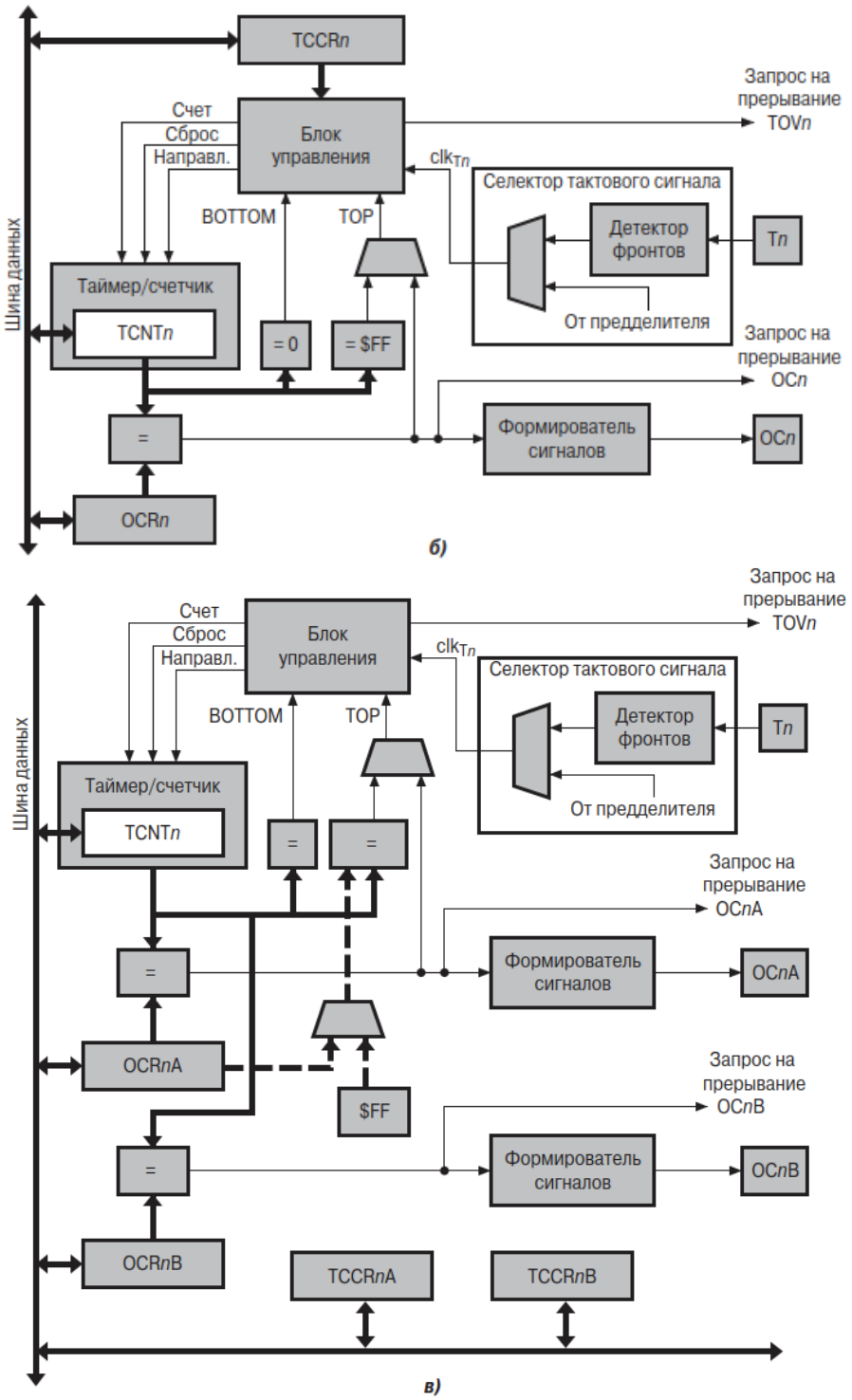


Рис. 2.10. Структурные схемы 8-битных таймеров-счетчиков без асинхронного режима

Таблица 2.9. **Функции таймера/счетчика T0**

Функции	ATmega8515x, ATmega8535x	ATmega8x	ATmega16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x	ATmega325x/3250x, ATmega645x/6450x	ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x
Восьмибитный счетчик	●	●	●	●	●	●	●	●	●	●
Счетчик внешних событий	●	●	●	—	●	●	●	●	●	●
Широтно-импульсный модулятор, число каналов	1	—	1	1	2	1	2	1	1	2
Часы реального времени	—	—	—	●	—	—	—	—	—	—

Таблица 2.10. **Функции таймера/счетчика T2**

Функции	ATmega8515x	ATmega8535x	ATmega8x	ATmega16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x	ATmega325x/3250x, ATmega645x/6450x	ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x
Восьмибитный счетчик	—	●	●	●	●	●	●	●	●	●	●
Счетчик внешних событий	—	—	—	—	●	—	—	—	—	—	—
Широтно-импульсный модулятор, число каналов	—	1	1	1	1	2	1	2	1	1	2
Часы реального времени	—	●	●	●	—	●	●	●	●	●	●

Количество регистров ввода/вывода, имеющихся в составе таймеров/счетчиков, зависит от сложности и возможностей последних. Все регистры 8-битных таймеров/счетчиков указаны в **Табл. 2.11**. Эти же факторы влияют и на количество прерываний, генерируемых конкретным таймером/счетчиком.

Регистр	Адрес	ATmega8515x	ATmega8535x	ATmega8x	ATmega16x/32x	64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x	ATmega325x/3250x, ATmega645x/6450x	ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x	Назначение
TCCR0	$33 ($53)	•	•	•	•	•		•					Регистр управления
TCCR0A	$24 ($44)						•		•	•	•	•	
TCCR0B	$25 ($45)						•		•			•	
TCNT0	$32 ($52)	•	•	•	•	•		•					Счетный регистр
	$26 ($46)						•		•	•	•	•	
OCR0	$31 ($51)	•				•		•					Регистр сравнения
	$3C ($5C)		•		•								
OCR0A	$27 ($47)						•		•	•	•	•	
OCR0B	$28 ($48)						•		•			•	
TCCR2	$25 ($45)		•	•	•	•							Регистр управления
	$27 ($47)							•					
TCCR2A	($B0)						•		•	•	•	•	
TCCR2B	($B1)						•		•			•	

Табл.2.11. Регистры 8-битных таймеров/счетчиков

(продолжение)

Регистр	Адрес	ATmega8515x	ATmega8535x	ATmega8x	ATmega16x/32x	64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x	ATmega325x/3250x, ATmega645x/6450x	ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x	Назначение
TCNT2	$24 ($44)		•	•	•	•							Счетный регистр
	$23 ($43)							•					
	($B2)						•		•	•	•	•	
OCR2	$23 ($43)		•	•	•	•							Регистр сравнения
OCR2A	($B3)						•		•	•	•	•	
OCR2B	($B4)						•		•			•	
ASSR	$22 ($42)		•	•	•								Регистр состояния асинхронного режима
	$26 ($46)							•					
	$30 ($50)					•							
	($B6)						•		•	•	•	•	

Счетный регистр таймера/счетчика TCNTn входит в состав основно- го блока модуля — блока реверсивного счетчика. В зависимости от режи- ма работы модуля содержимое счетного регистра сбрасывается, инкре- ментируется или декрементируется по каждому импульсу тактового сиг- нала таймера/счетчика clkT0 (clkT2). 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 опера- ция записи в счетный регистр блокирует работу блока сравнения на вре- мя одного периода тактового сигнала таймера/счетчика. После подачи напряжения питания в регистре TCNTn находится нулевое значение. При достижении таймером/счетчиком максимального или минимально- го значения (конкретный вариант зависит от его режима работы) уста навливается флаг TOVn в регистре флагов TIFR (TIFRn). Разрешение прерывания осуществляется установкой в 1 бита TOIEn регистра маски TIMSK (TIMSKn). Разумеется, флаг I регистра SREG также должен быть установлен в 1.

Регистры сравнения OCRn (OCRnA/OCRnB) входят в состав бло- ков сравнения модуля. Во время работы таймера/счетчика производит- ся непрерывное (в каждом такте) сравнение этих регистров с регистром TCNTn. В случае равенства содержимого этих регистров в следующем такте устанавливается флаг OCFn (OCFnA/OCFnB) в соответствующем регистре флагов и генерируется прерывание (если оно разрешено). Кроме того, при

наступлении этого события может изменяться состоя- ние вывода OCn (OCnA/OCnB) микроконтроллера. Чтобы тай- мер/счетчик мог управлять состоянием этих выводов, они должны быть сконфигурированы как выходы (соответствующий бит регистра DDRx должен быть установлен в 1).

Регистры TCCRn (TCCRnA/TCCRnB) предназначены для управ- ления модулем таймера/счетчика. Формат этих регистров приведен на Рис. 2.11, а описание их битов — соответственно в Табл. 2.12…2.13.

<table>
<tr><th>Бит</th><th>Название</th><th colspan="4">Описание</th></tr>
<tr><td>7</td><td>FOCn</td><td colspan="4">Принудительное изменение состояния вывода OCn (режимы Normal и CTC). При записи лог. 1 в этот бит состояние вывода OCn изменяется в соответствии с установками битов COMn1:COMn0. 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</td></tr>
<tr><td rowspan="5">6, 3</td><td rowspan="5">WGMn1:
WGMn0</td><td colspan="4">Режим работы таймера/счетчика. Эти биты определяют режим работы таймера/счетчика следующим образом:</td></tr>
<tr><td>Номер режима</td><td>WGMn1</td><td>WGMn0</td><td>Режим работы таймера/счетчика Tn</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Normal</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Phase correct PWM</td></tr>
<tr><td>2</td><td>1</td><td>0</td><td>CTC (сброс при совпадении)</td></tr>
<tr><td>5, 4</td><td>COMn1:
COMn0</td><td colspan="4">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 таймера/счетчика</td></tr>
<tr><td>2...0</td><td>CSn2...CSn0</td><td colspan="4">Управление тактовым сигналом. Эти биты определяют источник тактового сигнала таймера/счетчика. Действие этих битов зависит от исполнения таймера/счетчика и будет описано ниже</td></tr>
</table>

Табл.2.12. Биты регистра TCCR0 (TCCR2)

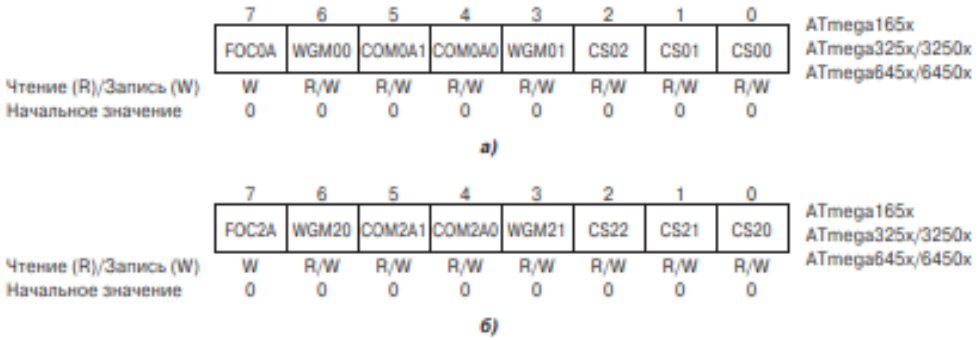


Рис. 2.11. Формат регистров TCCR0A (а) и TCCR2A (б) моделей Atmega165x/325x/3250x/645x/6450x

Таблица 2.13. **Биты регистров TCCR0A/TCCR0B (TCCR2A/TCCR2B)**

<table>
<tr><th>Регистр</th><th>Бит</th><th>Название</th><th>Описание</th></tr>
<tr><td rowspan="11">TCCRnA</td><td>7, 6</td><td>COMnA1: COMnA0</td><td colspan="5">Режим работы блока сравнения «A». Эти биты определяют поведение вывода OCnA при наступлении события «Совпадение». Влияние со- держимого этих битов на состояние вывода зависит от режима рабо- ты таймера/счетчика</td></tr>
<tr><td>5, 4</td><td>COMnB1: COMnB0</td><td colspan="5">Режим работы блока сравнения «B». Эти биты определяют поведение вывода OCnB при наступлении события «Совпадение». Влияние со- держимого этих битов на состояние вывода зависит от режима рабо- ты таймера/счетчика</td></tr>
<tr><td>3, 2</td><td>—</td><td colspan="5">Зарезервированы, читаются как 0</td></tr>
<tr><td rowspan="9">1, 0</td><td rowspan="9">WGMn1: WGMn0</td><td colspan="5">Режим работы таймера/счетчика. Эти биты совместно с битом WGMn2 регистра TCCRnB определяют режим работы таймера/счет- чика следующим образом:</td></tr>
<tr><td>Номер режима</td><td>WGMn2</td><td>WGMn1</td><td>WGMn0</td><td>Режим работы таймера/счетчика Tn</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>Normal</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>Phase correct PWM</td></tr>
<tr><td>2</td><td>0</td><td>1</td><td>0</td><td>CTC (сброс при совпадении)</td></tr>
<tr><td>3</td><td>0</td><td>1</td><td>1</td><td>Fast PWM</td></tr>
<tr><td>4</td><td>1</td><td>0</td><td>0</td><td>Зарезервировано</td></tr>
<tr><td>5</td><td>1</td><td>0</td><td>1</td><td>Phase correct PWM</td></tr>
<tr><td>6</td><td>1</td><td>1</td><td>0</td><td>Зарезервировано</td></tr>
<tr><td>7</td><td>1</td><td>1</td><td>1</td><td>Fast PWM</td></tr>
<tr><td rowspan="5">TCCRnB</td><td>7</td><td>FOCnA</td><td colspan="5">Принудительное изменение состояния вывода OCnA (режимы Normal и CTC). При записи лог. 1 в этот бит состояние вывода OCnA изменяет- ся в соответствии с установками битов COMnA1:COMnA0. Прерыва- ние 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</td></tr>
<tr><td>6</td><td>FOCnB</td><td colspan="5">Принудительное изменение состояния вывода OCnB (режимы Normal и CTC). При записи лог. 1 в этот бит состояние вывода OCnB изменяет- ся в соответствии с установками битов COMnB1:COMnB0. Прерыва- ние 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</td></tr>
<tr><td>5, 4</td><td>—</td><td colspan="5">Зарезервированы, читаются как 0</td></tr>
<tr><td>3</td><td>WGMn2</td><td colspan="5">Режим работы таймера/счетчика. Этот бит совместно с битами WGMn1:WGMn0 регистра TCCRnA определяют режим работы тайме- ра/счетчика</td></tr>
<tr><td>2…0</td><td>CSn2…CSn0</td><td colspan="5">Управление тактовым сигналом. Эти биты определяют источник так- тового сигнала таймера/счетчика. Действие этих битов зависит от ис- полнения таймера/счетчика и будет описано ниже</td></tr>
</table>

Примечание. *n* = 0 или 2.

Вопросы по 2 главе:

1. Какие три регистра связаны с каждым портом ввода/вывода (PORTx, DDRx, PINx) и какую функцию они выполняют?
2. Объясните, почему при чтении состояния вывода после его программного изменения необходимо вставлять команду NOP.
3. Какие источники тактового сигнала могут быть выбраны для 8-битного таймера/счётчика T0 с помощью битов CS02…CS00?
4. Перечислите основные режимы работы 8-битного таймера/счётчика (Normal, CTC, Fast PWM, Phase Correct PWM). В чём отличие CTC от Normal?
5. Что такое предделитель таймера и как сбросить его счётчики с помощью битов PSR (в регистрах SFIOR/GTCCR)?
6. Какие флаги и маски прерываний используются для обработки событий переполнения и совпадения для таймера T0 (регистры TIFR0, TIMSK0)?
7. В чём заключается двойная буферизация регистров сравнения OCR0A/OCR0B в режимах ШИМ и зачем она нужна?
8. Какой внешний сигнал может тактировать таймер T0? Какие ограничения по частоте и как выбирается фронт счёта?

Глава 3. Микроконтроллеры

3.1. Управление тактовым сигналом

Формирование тактового сигнала таймера/счетчика clkT0 (clkT2) осуществляется блоком предделителя. В качестве тактового сигнала clkT0 (clkT2) таймеров/счетчиков, не имеющих асинхронного режима, может использоваться:

- системный тактовый сигнал (clkT0 (T2) = clkI/O);
- масштабированный системный тактовый сигнал (clkT0 (T2) = clkI/O/n);
- внешний сигнал, поступающий на вход T0 (T2) микроконтроллера (clkT0 (T2) = clkEXT).

Тактовый сигнал таймеров/счетчиков с асинхронным режимом может формироваться либо из системного тактового сигнала clkI/O (clkT0 (T2) = clkI/O/n), либо — в асинхронном режиме — из сигнала от допол- нительного кварцевого резонатора (clkT0 (T2) = clkTOSC1/n). Переключение между синхронным и асинхронным режимами работы осуществляется с помощью бита AS0 (AS2) регистра ASSR.

Выбор источника тактового сигнала, а также запуск и остановка таймеров/счетчиков осуществляются с помощью битов CS02…CS00 (CS22…CS20) регистров управления таймером TCCRn (TCCRnA/TCCRnB) согласно Табл.3.1.

Таблица 3.1. **Выбор источника тактового сигнала таймеров/счетчиков T0 и T2**

CS*n*2	CS *n*1	CS *n*0	Источник тактового сигнала		
			Обычный таймер/счетчик	Асинхронный таймер/счетчик	
				AS*n* = 0	AS*n* = 1
0	0	0	Таймер/счетчик остановлен	Таймер/счетчик остановлен	
0	0	1	$clk_{I/O}$	$clk_{I/O}$	clkTOSC1
0	1	0	$clk_{I/O}/8$	$clk_{I/O}/8$	clkTOSC1/8
0	1	1	$clk_{I/O}/64$	$clk_{I/O}/32$	$clk_{TOSC1}/32$
1	0	0	$clk_{I/O}/256$	$clk_{I/O}/64$	$clk_{TOSC1}/64$
1	0	1	$clk_{I/O}/1024$	$clk_{I/O}/128$	$clk_{TOSC1}/128$
1	1	0	Вывод T*n*, счет осуществляется по спадающему фронту импульсов	$clk_{I/O}/256$	$clk_{TOSC1}/256$
1	1	1	Вывод T*n*, счет осуществляется по нарастающему фронту импульсов	$clk_{I/O}/1024$	$clk_{TOSC1}/1024$

Примечание. *n* = 0 или 2.

Режимы работы

Режим работы таймера/счетчика T0 (T2) определяется состоянием би- тов WGM*n*2:WGM*n*0 регистра TCCR*n* (TCCR*n*A/TCCR*n*B). Зависимость режима работы таймеров/счетчиков от состояния этих битов показана в **Табл. 3.2**.

Таблица 3.2. **Режимы работы таймеров/счетчиков T0 и T2**

Номер режима	WGM*n*2[1)]	WGM*n*1	WGM*n*0	Режим работы таймера/счетчика T*n*	Модуль счета (TOP)	Обновление регистров OCR*nx*	Момент установки флага TOV*n*
0	0	0	0	Normal	$FF	Немедленно	$FF
1	0	0	1	Phase correct PWM	$FF	При TOP	$00
2	0	1	0	CTC (сброс при совпадении)	OCR*n* (OCR*n*A)	Немедленно	$FF
3	0	1	1	Fast PWM	$FF	При TOP	$FF
4[1)]	1	0	0	Зарезервировано	—	—	—
5[1)]	1	0	1	Phase correct PWM	OCR*n* (OCR*n*A)	При TOP	$00
6[1)]	1	1	0	Зарезервировано	—	—	—
7[1)]	1	1	1	Fast PWM	OCR*n* (OCR*n*A)	При TOP	TOP

[1)] В моделях ATmega48x/88x/168x, ATmega164x/324x/644x и ATmega640x/1280x/1281x/2560x/2561x.

Примечание. *n* = 0 или 2.

Режим Normal

Это наиболее простой режим работы таймеров/счетчиков. А в таймере/счетчике T0 модели ATmega8x это вообще единственный режим. В ре- жиме Normal счетный регистр функционирует как обычный суммирую- щий счетчик. По каждому импульсу тактового сигнала clkTn осуществляет- ся инкрементирование счетного регистра. При переходе через значение

$FF возникает переполнение, и счет продолжается со значения $00. В том же такте сигнала clkTn, в котором обнуляется регистр TCNTn, флаг прерывания по переполнению TOVn устанавливается в 1.

При равенстве счетного регистра и регистра сравнения устанавливает- ся соответствующий флаг прерывания OCFn (OCFnA/OCFnB) и, если бит OCIEn (OCIEnA/OCIEnB) регистра маски установлен в 1, генерируется прерывание. Наряду с установкой флага при равенстве счетного регистра и регистра сравнения может изменяться состояние вывода OCn (OCnA/OCnB) микроконтроллера. Каким образом оно будет изменяться, определяется битами COMn1:COMn0 (COMnx1:COMnx0) регистра управ- ления TCCRn/TCCRnA в соответствии с Табл. 3.3.

Таблица 3.3. **Управление выводами OC*n* (OC*n*A/OC*n*B) в режиме Normal**

COM*n*1 (COM*nx*1)	COM*n*0 (COM*nx*0)	Описание
0	0	Таймер/счетчик T*n* отключен от вывода OC*n* (OC*nx*)
0	1	Состояние вывода меняется на противоположное
1	0	Вывод сбрасывается в 0
1	1	Вывод устанавливается в 1

Примечание. *n* = 0 или 2.

При необходимости состояние вывода OCn (OCnA/OCnB) может быть изменено принудительно записью лог. 1 в бит FOCn (FOCnA/FOCnB) соответствующего регистра управления. Прерывание при этом не генерируется.

Режим CTC (сброс при совпадении)

В этом режиме счетный регистр тоже функционирует как обычный суммирующий счетчик, инкрементирование которого осуществляется по каждому импульсу тактового сигнала clkTn. Однако максимально возмож- ное значение счетного регистра и, следовательно, разрешающая способ- ность счетчика определяются регистром сравнения OCRn (OCRnA). После достижения значения, записанного в регистре сравнения, счет продолжа- ется со значения $00. Если в регистре сравнения записано $FF, то в том же такте сигнала clkTn, в котором обнуляется счетный регистр, устанавливает- ся флаг прерывания по переполнению TOVn в соответствующем регистре флагов. Временные диаграммы для этого режима работы таймера/счетчи- ка приведены на Рис. 3.1.

При достижении счетчиком максимального значения устанавливается флаг OCFn (OCFnA), и, если бит OCIEn (OCIEnA) соответствующего ре- гистра маски установлен в 1, генерируется прерывание. Одновременно с установкой флага может изменяться состояние выводов OCn (OCnA/OCnB) микроконтроллера. Состояние выводов определяется бита- ми COMn1:COMn0 (COMnx1:COMnx0) регистра управления TCCRn/TCCRnA, как указано в Табл. 3.4.

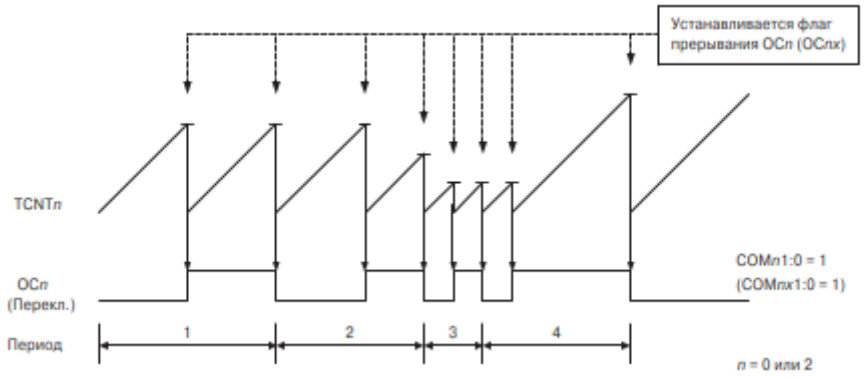


Рис.3.1. Временные диаграммы для режима CTC

Таблица 3.4. **Управление выводами OC*n* (OC*n*A/OC*n*B) в режиме CTC**

COM*n*1 (COM*nx*1)	**COM*n*0 (COM*nx* 0)**	**Описание**
0	0	Таймер/счетчик T*n* отключен от вывода OC*n* (OC*nx*)
0	1	Состояние вывода меняется на противоположное
1	0	Вывод сбрасывается в 0
1	1	Вывод устанавливается в 1

Примечание. *n* = 0 или 2.

Для генерации сигнала заданной частоты необходимо записать в биты COM*n*1:COM*n*0 (COM*nx*1:COM*nx*0) значение 01 (переключение вывода). Частота генерируемого сигнала будет определяться выражением

$$f_{OCn} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCR_{nx})},$$

где N — коэффициент деления предделителя (см. Табл. 3.1).

При необходимости состояние вывода OC*n* (OC*n*A/OC*n*B) можно изменить принудительно записью лог. 1 в бит FOC*n* (FOC*n*A/FOC*n*B) соответствующего регистра управления. Прерывание при этом не генерируется и сброса счетного регистра не производится.

Режим Fast PWM

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

Счетный регистр в этом режиме функционирует как суммирующий счетчик, инкрементирование которого осуществляется по каждому им- пульсу тактового сигнала clk_{Tn}. 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 регист- ра сравнения OCR*n* (OCR*n*A/ OCR*n*B) устанавливается флаг OCF*n* (OCF*n*A/OCF*n*B). Максимальное значение равно $FF (при WGM*n*2 = 0 или при отсутствии этого бита в регистре микроконтроллера) или задается регистром OCR*n*A (при WGM*n*2 = 1).

Особенностью работы схемы сравнения в этом режиме является двой- ная буферизация записи в регистр OCR*n* (OCR*n*A/OCR*n*B), которая за- ключается в том, что записываемое число на самом деле сохраняется в спе- циальном буферном регистр, а изменение содержимого регистра сравне- ния происходит только в момент достижения счетчиком максимального значения. Благодаря такому решению исключается появление несиммет- ричных импульсов сигнала (помех) на выходе модулятора, которые были бы неизбежны при непосредственной записи в регистр сравнения.

Состояние выводов OC*n* (OC*n*A/OC*n*B) микроконтроллера в этом режиме также определяется содержимым битов COM*n*1:COM*n*0 (COM*nx*1:COM*nx*0) регистра TCCR*n*/TCCR*n*A (см. **Табл. 3.5 и рис. 3.2).**

Таблица 3.5. **Управление выводами OC*n* (OC*n*A/OC*n*B) в режиме Fast PWM**

COM*n*1 (COM*nx*1)	**COM*n*0 (COM*nx*0)**	**Описание**
0	0	Таймер/счетчик T*n* отключен от вывода OC*n* (OC*nx*)
0	1	**OC*n*A:** WGM*n*2 = 0 — таймер/счетчик T*n* отключен от вывода OC*n*A; WGM*n*2 = 1 — состояние вывода меняется на противоположное при равенстве регистров TCNT*n* и OCR*n*A. **OC*n*B, OC*n*C:** Зарезервировано
1	0	Сбрасывается в 0 при равенстве регистров TCNT*n* и OCR*n* (OCR*nx*). Устанавливается в 1 при достижении счетчиком максимального значе- ния (неинвертированный ШИМ-сигнал)
1	1	Устанавливается в 1 при равенстве регистров TCNT*n* и OCR*n* (OCR*nx*). Сбрасывается в 0 при достижении счетчиком максимального значения (инвертированный ШИМ-сигнал)

Примечание. *n* = 0 или 2.

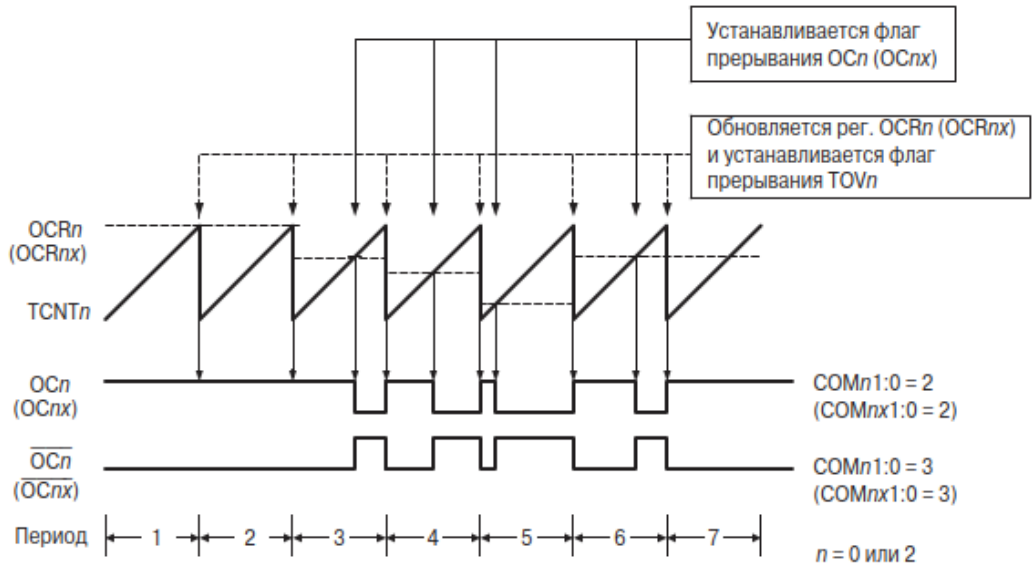


Рис.3.2. Формирование ШИМ-сигнала в режиме Fast PWM

Отдельно следует рассматривать случаи, когда в регистре сравнения находятся предельно возможные значения. Если в регистре сравнения OCR*n* (OCR*n*A/OCR*n*B) содержится $00, то на выходе OC*n* (OC*n*A/OC*n*B) будут наблюдаться короткие выбросы с периодом, равным максимальному значению таймера/счетчика. Если же содержимое регистра сравнения рав- но максимальному значению, то вывод OC*n* (OC*n*A/OC*n*B) переключится в устойчивое состояние, определяемое установками битов COM*n*1:COM*n*0 (COM*nx*1:COM*nx*0). В частности, подобным образом функционирует вы- ход OC*n*A, когда максимальное значение задается регистром OCR*n*A.

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.

Максимальное значение равно $FF (при WGMn2 = 0 или при отсут- ствии этого бита в регистре микроконтроллера) или задается регистром OCRnA (при WGMn2 = 1). При достижении счетчиком максимального значения происходит смена направления счета, однако счетчик остается в этом состоянии в течение одного периода сигнала clkTn. При достижении счетчиком минимального значения ($00) также происходит смена направ- ления счета и одновременно устанавливается флаг прерывания TOVn в со- ответствующем регистре флагов. При равенстве содержимого счетного ре- гистра и регистра сравнения OCRn (OCRnA/OCRnB) устанавливается флаг OCFn (OCFnA/OCFnB) и изменяется состояние вывода OCn (OCnA/OCnB). Характер изменения определяется, как обычно, содержи- мым битов COMn1:COMn0 (COMnx1:COMnx0) регистра TCCRn/TCCRnA (Табл. 3.6 и Рис. 3.3).

COM*n*1 (COM*nx*1)	**COM*n*0 (COM*nx*0)**	**Описание**
0	0	Таймер/счетчик T*n* отключен от вывода OC*n* (OC*nx*)
0	1	**OC*n*A:** WGM*n*2 = 0 — таймер/счетчик T*n* отключен от вывода OC*n*A; WGM*n*2 = 1 — состояние вывода меняется на противоположное при равенстве регистров TCNT*n* и OCR*n*A. **OC*n*B, OC*n*:** Зарезервировано
1	0	Сбрасывается в 0 при прямом счете и устанавливается в 1 при обратном счете (неинвертированный ШИМ-сигнал)
1	1	Устанавливается в 1 при прямом счете и сбрасывается в 0 при обратном счете (инвертированный ШИМ-сигнал)

Примечание. *n* = 0 или 2.

Табл. 3.6 Управление выводами OCn (OCnA/OCnB)
в режиме Phase Correct PWM

Для исключения несимметричных выбросов в этом режиме тоже реализована двойная буферизация записи в регистры сравнения. Поэтому действительное изменение содержимого регистра сравнения происходит только в момент достижения счетчиком максимального значения.

Если в регистр сравнения записать минимальное ($00) или максималь- ное значение, то при следующем совпадении состояния счетчика и содержимого регистра сравнения выход схемы сравнения переключится в устойчивое состояние согласно **Табл.3.7.**

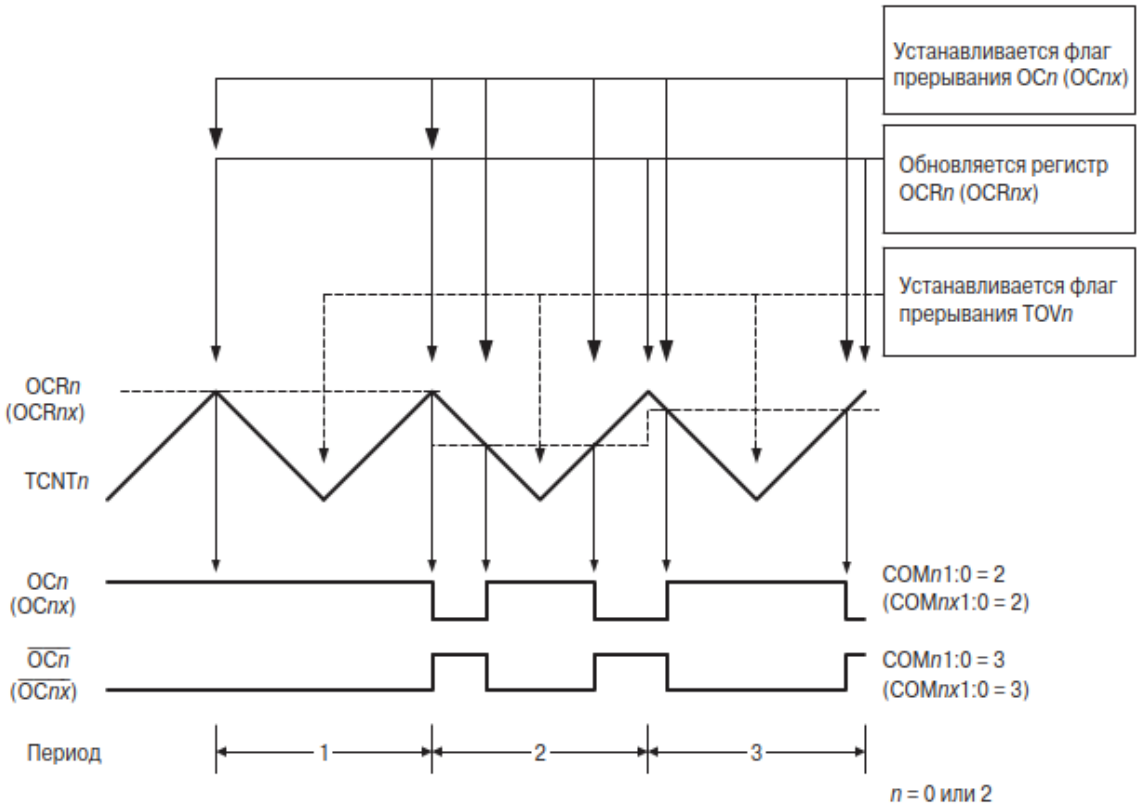


Рис.3.3. Формирование ШИМ-сигнала в режиме Phase Correct PWM

3.2. Асинхронный режим

В моделях ATmega64x и ATmega128x в асинхронном режиме может работать таймер/счетчик T0. В остальных моделях (кроме ATmega8515x) такой возможностью обладает таймер/счетчик T2.

В асинхронном режиме на вход предделителя поступает сигнал от кварцевого генератора таймера/счетчика, что позволяет использовать таймер/счетчик в качестве часов реального времени. В качестве источника сигнала, как правило, используется кварцевый резонатор, подключаемый к выводам TOSC1 и TOSC2 микроконтроллера. В некоторых моделях можно использовать сигнал от внешней схемы, подаваемый на вывод TOSC1. Тактовый генератор таймера/счетчика оптимизирован для работы на частоте 32 768 Гц, при этом она должна быть как минимум в 4 раза ниже частоты тактового сигнала микроконтроллера.

Непосредственная запись в регистры TCNT*n*, OCR*n* (OCR*n*A/OCR*n*B) и TCCR*n* (TCCR*n*A/TCCR*n*B) 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 управления асинхронным режимом таймера/счетчика предназначен регистр ASSR. Формат этого регистра приведен на Рис. 3.7, а описание его битов — в Табл.3.4.

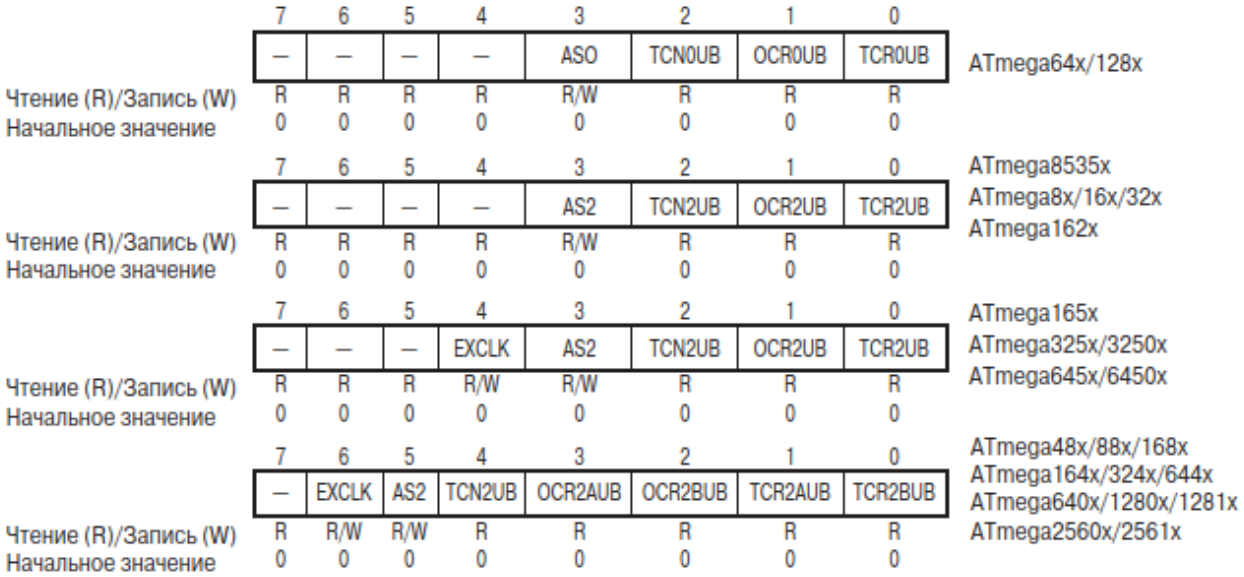


Рис.3.7. Формат регистра ASSR

Таблица 3.4. **Биты регистра ASSR**

Название	Описание
AS*n*	**Переключение режима работы.** Если бит установлен в 1, то на вход предделителя таймера/счетчика T*n* поступают импульсы с кварцевого генератора таймера/счет- чика (асинхронный режим). В этом режиме выводы TOSC1 и TOSC2 используются для подключения кварцевого резонатора и соответственно не могут использоваться как контакты ввода/вывода общего назначения. Если бит сброшен в 0, то на вход предделителя поступает внутренний тактовый сигнал микроконтроллера. В этом случае выводы TOSC1 и TOSC2 являются линия- ми ввода/вывода общего назначения. При изменении состояния этого бита содержимое регистров TCNT*n*, OCR*n* (OCR*n*A/OCR*n*B) и TCCR*n* (TCCR*n*A/TCCR*n*B) может быть повреждено
EXCLK	**Разрешение внешнего тактового сигнала.** При установке бита в 1 включается входной буфер внешнего тактового сигнала и таймер/счетчик может тактироваться внешним сигналом, подаваемым на вход TOSC1. Установка этого бита должна про- изводиться перед включением асинхронного режима. Если в регистре ASSR конкретной модели этот бит отсутствует, то использование внешнего тактового сигнала с этим микроконтроллером не рекомендуется
TCN*n*UB	**Состояние обновления регистра TCNT*n*.** При записи в регистр TCNT*n* этот флаг ус- танавливается в 1, а после пересылки записываемого значения в данный регистр флаг аппаратно сбрасывается в 0. Таким образом, сброшенный флаг TCN*n*UB озна- чает, что регистр TCNT*n* готов для записи в него нового значения. Запись в регистр TCNT*n* при установленном флаге TCN*n*UB может привести к пов- реждению прежнего содержимого регистра и к генерации прерывания
OCR*n*UB, OCR2AUB, OCR2BUB	**Состояние обновления регистра OCR*n* (OCR2A/OCR2B).** При записи в регистр срав- нения соответствующий флаг устанавливается в 1, а после пересылки записываемо- го значения в регистр флаг аппаратно сбрасывается в 0. Таким образом, сброшен- ный флаг OCR*n*UB (OCR2AUB/OCR2BUB) означает, что соответствующий регистр сравнения готов для записи в него нового значения. Запись в регистр сравнения при установленном флаге OCR*n*UB (OCR2AUB/OCR2BUB) может привести к повреждению прежнего содержимого регистра и к генерации прерывания

TCR*n*UB, TCR2AUB, TCR2BUB	**Состояние обновления регистра TCCR*n* (TCCR2A/TCCR2B).** При записи в регистр управления соответствующий флаг устанавливается в 1, а после пересылки записы- ваемого значения в регистр флаг аппаратно сбрасывается в 0. Таким образом, сбро- шенный флаг TCR*n*UB (TCR2AUB/TCR2BUB) означает, что соответствующий ре- гистр управления готов для записи в него нового значения. Запись в регистр управления при установленном флаге TCR*n*UB (TCR2AUB/TCR2BUB) может привести к повреждению прежнего содержимого регистра и к генерации прерывания

Примечание. *n* = 0 для моделей ATmega64x/128x и 2 — для остальных моделей (кроме ATmega8515x)

Необходимо отметить, что при переключении между синхронным и асинхронным режимами содержимое регистров таймера/счетчика может быть повреждено. Чтобы этого избежать, рекомендуется придерживаться следующей последовательности действий:

1. Запретить прерывания от таймера/счетчика.
2. Переключить его в требуемый режим.
3. Записать новые значения в регистры TCNTn, OCRn (OCRnx) и TCCRn (TCCRnx).
4. В случае переключения в асинхронный режим дождаться сброса флагов TCNnUB, OCRnUB (OCR2xUB) и TCRnUB (TCR2xUB).
5. Сбросить флаги прерываний таймера/счетчика.
6. Разрешить прерывания (если требуется).

При работе таймера/счетчика в асинхронном режиме установка флагов прерываний от него производится синхронно с тактовым сигналом микроконтроллера. Для синхронизации требуется 3 такта плюс один период тактового сигнала таймера/счетчика. Поэтому к моменту, когда микроконтроллер сможет прочитать состояние счетчика, вызвавшее установку флага прерывания, оно изменится, по меньшей мере, на единицу. Изменение состояния выводов OCn (OCnA/OCnB) производится по тактовому сигналу таймера/счетчика и не синхронизируется с тактовым сигналом микроконтроллера. Отдельно следует сказать о «взаимодействии» асинхронного режима таймеров/счетчиков с режимами пониженного энергопотребления микроконтроллера.

Первое замечание касается использования прерываний от тайме- ра/счетчика для «пробуждения» микроконтроллера. Если перевод микро- контроллера в режим Power Save или Extended Standby осуществляется сра- зу же после записи в регистры таймера/счетчика, необходимо убедиться, что операция записи завершена. Наиболее важно это в случае, когда для «пробуждения» микроконтроллера используется прерывание от блока сравнения, поскольку во время записи в счетный регистр или регистр сравнения работа блока сравнения заблокирована. Соответственно, если переход в «спящий» режим произойдет до окончания операции записи в указанные регистры, прерывания от схемы сравнения никогда не про- изойдет, и микроконтроллер не сможет выйти из «спящего» режима.

Кроме того, необходимо быть осторожным при повторном переходе в режим Power Save или Extended Standby после выхода из них по прерыванию от

таймера/счетчика. Дело в том, что в этом случае для запуска подсистемы прерываний требуется промежуток времени, равный одному периоду сигнала на выводе TOSC1. Если же промежуток времени между «пробуждением» и повтор- ным переходом в «спящий» режим будет меньше указанного, генерации преры- вания и соответственно перехода микроконтроллера в рабочий режим не про- изойдет. Для формирования задержки требуемой длительности рекомендуется после «пробуждения» микроконтроллера выполнить запись в какой-либо из регистров таймера/счетчика и дождаться завершения этой операции.

После подачи напряжения питания, а также после «пробуждения» микроконтроллера из режима Power Down или Standby таймер/счетчик рекомендуется использовать только спустя секунду после указанных собы- тий. Эта задержка необходима для запуска тактового генератора таймера/счетчика. Соответственно, при выходе из режима Power Down или Standby содержимое всех регистров таймера/счетчика можно считать поте- рянным (из-за нестабильности тактового сигнала во время запуска генера- тора). Причем это справедливо не только при использовании кварцевого резонатора, но и при использовании внешнего тактового сигнала.

16-битные таймеры/счетчики

Количество 16-битных таймеров/счетчиков зависит от модели. Таймер/счетчик T1 присутствует во всех моделях микроконтроллеров семей- ства Mega. В моделях ATmega64x/128x и ATmega162x имеется уже два 16-битных таймера/счетчика — T1 и T3. Больше всего 16-битных тайме- ров/счетчиков в моделях ATmega640x/1280x/1281x/2560x/2561x — целых пять (T1, T3…T5). Как и 8-битные таймеры/счетчики T0 и T2, они могут использоваться для формирования временных интервалов, для подсчета числа внешних событий, формирования сигналов и генерации сигналов с ШИМ. В дополнение к этому 16-битные таймеры/счетчики могут по внешнему сигналу сохранять свое текущее состояние в отдельном регистре ввода/вывода.

Рассматриваемые таймеры/счетчики различных моделей отличаются только количеством блоков сравнения и соответственно количеством ка- налов генерации ШИМ-сигналов. Так, если в моделях ATmega64x/128x и ATmega640x/1280x/1281x/2560x/2561x 16-битные таймеры/счетчики име- ют по три блока сравнения, то в остальных моделях — только по два. Кро- ме того, таймеры/счетчики T4 и T5 моделей ATmega1281x/2561x являются усеченными — из-за малого количества выводов микроконтроллера в них не поддерживаются функции захвата и сравнения, а также тактирование от внешнего сигнала. Упрощенная структурная схема 16-битных таймеров/счетчиков приведена на Рис. 3.8.

В состав каждого таймера/счетчика входят следующие регистры ввода/вывода:

- 16-битный счетный регистр TCNTn;

- 16-битный регистр захвата ICRn;
- два или три 16-битных регистра сравнения OCRnA, OCRnB, OCRnC;
- два или три 8-битных регистра управления TCCRnA, TCCRnB, TCCRnC.

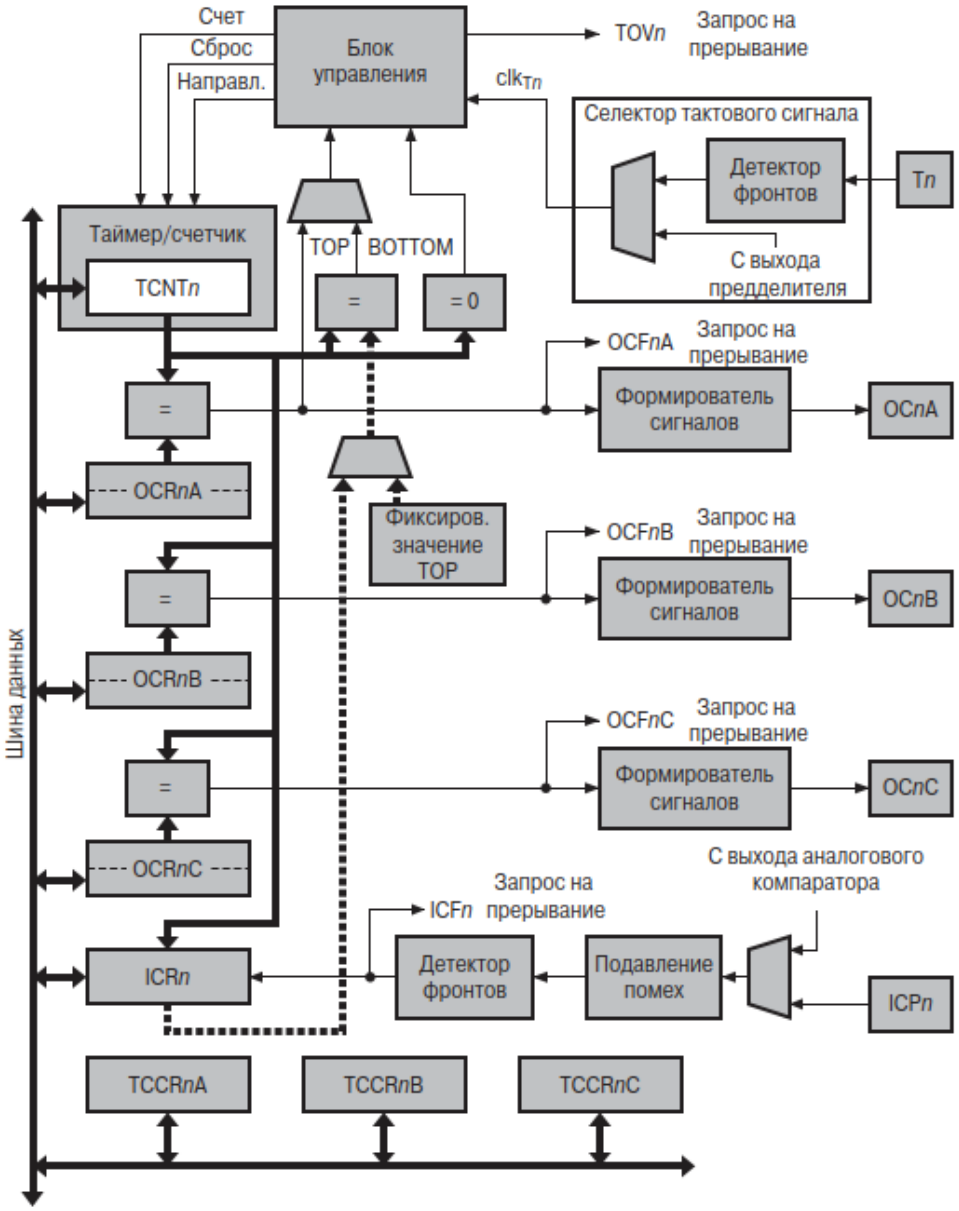


Рис.3.8. Структурная схема 16-битных таймеров/счетчиков (T1, T3, T4, T5)

Каждый из 16-битных регистров физически размещается в двух регистрах ввода/вывода, названия которых получаются добавлением к названию регистра буквы «H» (старший байт) и «L» (младший байт). Счетный ре- гистр таймера/счетчика T1 TCNT1, например, размещается в регистрах TCNT1H:TCNT1L. Адреса всех регистров 16-битных таймеров/счетчиков указаны в Табл. 3.5.

Регистр	Адрес	ATmega8515x	ATmega8535x	ATmega8x	ATmega16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x	ATmega325x/3250x, ATmega645x/6450x	ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x
TCCR1A	$2F ($4F)	•	•	•	•	•		•				
	($80)						•		•	•	•	•
TCCR1B	$2E ($4E)	•	•	•	•	•		•				
	($81)						•		•	•	•	•
TCCR1C	($7A)					•						
	($82)						•		•	•	•	•
TCNT1	$2D:$2C ($4D:$4C)	•	•	•	•	•		•				
	($85:$84)						•		•	•	•	•
OCR1A	$2B:$2A ($4B:$4A)	•	•	•	•	•		•				
	($89:$88)						•		•	•	•	•
OCR1B	$29:$28 ($49:$48)	•	•	•	•	•		•				
	($8B:$8A)						•		•	•	•	•
OCR1C	($79:$78)					•						
	($8D:$8C)											•
ICR1	$27:$26 ($47:$46)		•	•	•	•						
	$25:$24 ($45:$44)	•						•				
	($87:$86)						•		•	•	•	•
TCCR3A	($8B)					•		•				
	($90)											•
TCCR3B	($8A)					•		•				
	($91)											•
TCCR3C	($8C)					•						
	($92)											•
TCNT3	($89:$88)					•		•				
	($95:$94)											•

Регистр	Адрес	ATmega8515x	ATmega8535x	ATmega8x	ATmega16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x	ATmega325x/3250x, ATmega645x/6450x	ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x
TCCR1A	$2F ($4F)	•	•	•	•	•		•				
	($80)						•		•	•	•	•
TCCR1B	$2E ($4E)	•	•	•	•	•		•				
	($81)						•		•	•	•	•
TCCR1C	($7A)					•						
	($82)						•		•	•	•	•
TCNT1	$2D:$2C ($4D:$4C)	•	•	•	•	•		•				
	($85:$84)						•		•	•	•	•
OCR1A	$2B:$2A ($4B:$4A)	•	•	•	•	•		•				
	($89:$88)						•		•	•	•	•
OCR1B	$29:$28 ($49:$48)	•	•	•	•	•		•				
	($8B:$8A)						•		•	•	•	•
OCR1C	($79:$78)					•						
	($8D:$8C)											•
ICR1	$27:$26 ($47:$46)		•	•	•	•						
	$25:$24 ($45:$44)	•						•				
	($87:$86)						•		•	•	•	•
TCCR3A	($8B)					•		•				
	($90)											•
TCCR3B	($8A)					•		•				
	($91)											•
TCCR3C	($8C)					•						
	($92)											•
TCNT3	($89:$88)					•		•				
	($95:$94)											•

Табл.3.5. Регистры 16-битных таймеров/счетчиков

Таймеры/счетчики T1 и T3…T5 могут генерировать прерывание при наступлении следующих событий:

- переполнение счетного регистра;
- равенство счетного регистра и регистра сравнения (по одному пре- рыванию на каждый блок сравнения);
- сохранение счетного регистра в регистре захвата.

Флаги всех прерываний 16-битных таймеров/счетчиков находятся в регистрах флагов TIFR/ETIFR/TIFRn, а разрешение/запрещение этих прерываний

осуществляется установкой/сбросом соответствующих фла- гов регистров TIMSK/ETIMSK/TIMSKn (см. раздел 7.3).

Счетный регистр таймера/счетчика TCNTn входит в состав основного блока модуля — блока реверсивного счетчика. В зависимости от режима работы модуля содержимое счетного регистра сбрасывается, инкрементиру- ется или декрементируется по каждому импульсу тактового сигнала тай- мера/счетчика clkTn. Независимо от того, присутствует тактовый сигнал или нет, регистр доступен в любой момент времени как для чтения, так и для записи. При этом любая операция записи в счетный регистр блокирует работу всех блоков сравнения на время одного периода тактового сигнала таймера/счетчика. После подачи напряжения питания в регистре TCNTn находится нулевое значение. При некоторых изменениях состояния тай- мера/счетчика, определяемых режимом его работы, устанавливается бит TOVn в соответствующем регистре флагов. Разрешение прерывания осу- ществляется установкой в 1 бита TOIEn соответствующего регистра маски. Регистры сравнения OCRnA/OCRnB/OCRnC входят в состав блоков сравнения. Во время работы таймера/счетчика производится непрерывное (в каждом такте) сравнение этих регистров с регистром TCNTn. В случае равенства содержимого регистра сравнения и счетного регистра в следую- щем такте устанавливается флаг OCFnA/OCFnB/OCFnC в соответствую- щем регистре флагов и генерируется прерывание (если оно разрешено). Также при наступлении этого события может изменяться состояние выво- да OCnA/OCnB/OCnC микроконтроллера. Чтобы таймер/счетчик мог уп- равлять состоянием какого-либо из этих выводов, он должен быть скон- фигурирован как выходной (соответствующий бит регистра DDRx долженбыть установлен в 1).

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

Регистр захвата ICRn входит в состав блока захвата, назначение кото- рого — сохранение в определенный момент времени состояния тайме- ра/счетчика в регистре захвата. Это действие может производиться либо по активному фронту сигнала на выводе ICPn микроконтроллера, либо (для таймера/счетчика T1) по сигналу от аналогового компаратора. Одновре- менно с записью в регистр захвата устанавливается флаг ICFn соответству- ющего регистра флагов и генерируется запрос на прерывание. Разрешение прерывания осуществляется установкой в 1 бита TICIEn регистра маски.

Программно запись в регистр ICRn возможна только в режимах, в которых регистр захвата определяет модуль счета таймера/счетчика. Вывод ICPn в этих режимах отключен от микроконтроллера, а функция захвата соответственно выключена.

Упрощенная структурная схема блока захвата приведена на Рис. 3.9.

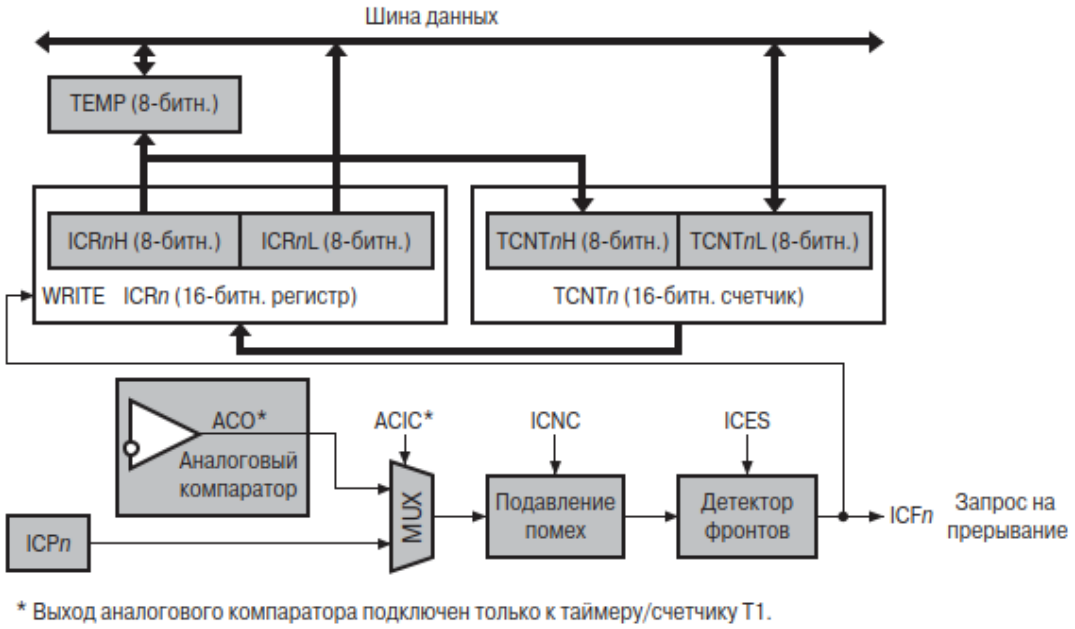


Рис.3.9. Структурная схема блока захвата

Для управления схемой захвата используются два бита регистра TCCR*n*B — ICNC*n* и ICES*n*. Бит ICNC*n* управляет схемой подавления по- мех. Если этот бит сброшен в 0, схема подавления помех выключена и за- хват производится по первому же активному фронту на выходе мульти- плексора (**Рис. 3.9**). Если же этот бит установлен в 1, то при появлении активного фронта производится 4 выборки с частотой, равной тактовой частоте микроконтроллера. Захват будет выполнен только в том случае, ес- ли все выборки имеют уровень, соответствующий активному фронту сиг- нала (лог. 1 для нарастающего и лог. 0 для спадающего).

Активный фронт сигнала, т. е. фронт, по которому будет выполнено со- хранение содержимого счетного регистра в регистре захвата, определяется битом ICES*n*. Если этот бит сброшен в 0, то активным является спадаю- щий фронт. Если бит установлен в 1, то активным является нарастающий фронт. Для захвата по сигналу с вывода ICP*n* этот вывод должен быть сконфигурирован как вход (бит регистра DDR*x*, соответствующий выводу, должен быть сброшен в 0). Если же он будет сконфигурирован как выход, захват можно будет осуществлять программно, управляя соответствующим битом порта.

Следует понимать, что между изменением состояния входа блока за- хвата и копированием счетного регистра в регистр захвата таймера/счетчи- ка проходит некоторое время. Эту задержку вносит каскад, состоящий из синхронизатора (на рисунке не показан) и детектора фронтов. Величина задержки составляет от 2.5 до 3.5 тактов. При включении схемы подавле- ния помех задержка увеличивается еще на 4 такта.

Для управления таймером/счетчиком используются три регистра уп- равления: TCCRnA, TCCRnB, TCCRnC. Формат этих регистров приведен на Рис.3.10 - 3.11, а описание их битов — в Табл. 3.6-3.7.

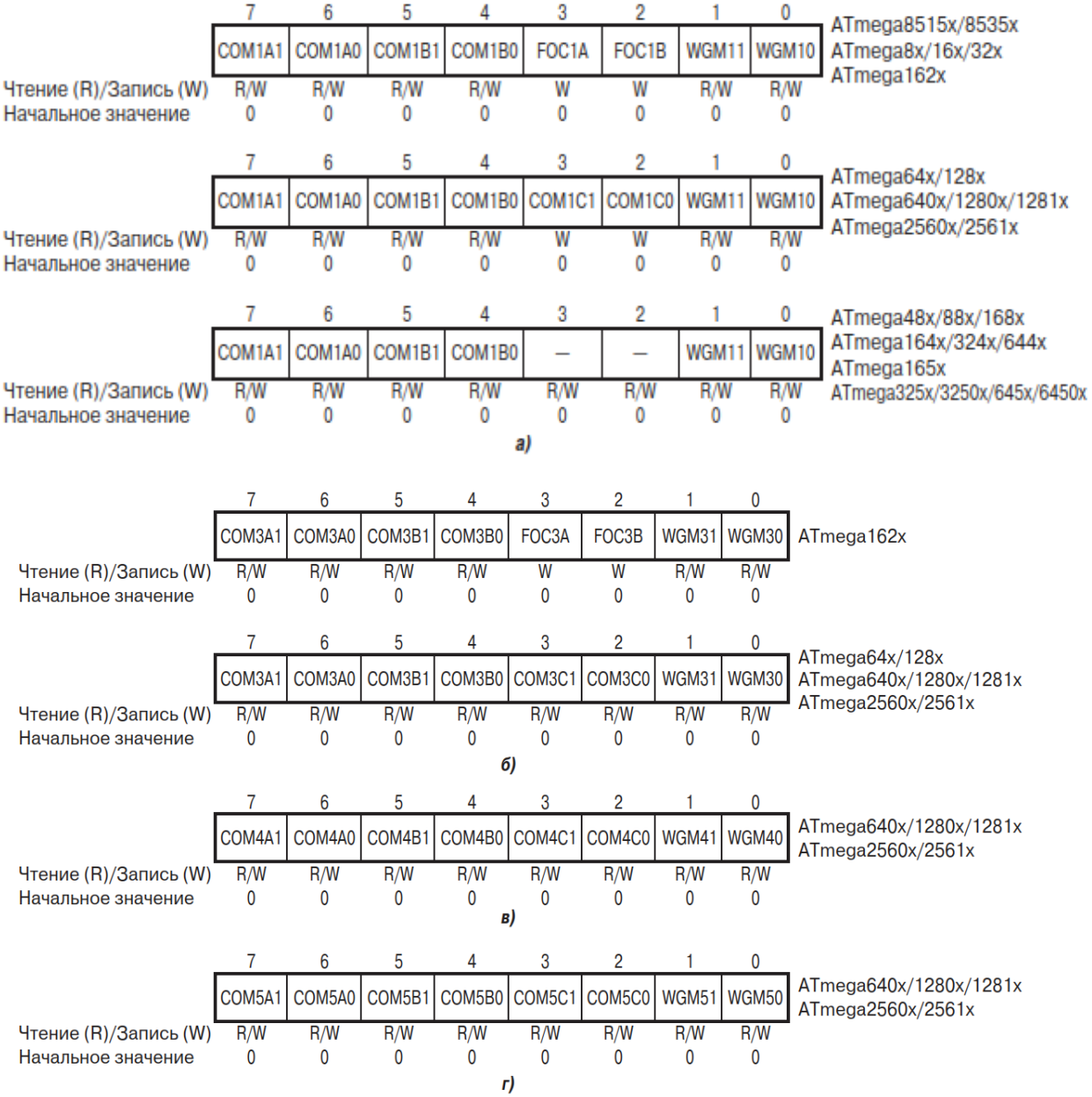


Рис.3.10. Формат регистров TCCR1A (а), TCCR3A (б), TCCR4A (в) и TCCR5A (г)

Таблица 3.6. **Биты регистров TCCR*n*A**

Название	Описание
COM*n*A1:COM*n*A0	**Режим работы блока сравнения *x*.** Эти биты определяют состояние вывода OC*nx* при наступлении события «Совпадение». Влияние содержимого этих битов на состояние вывода зависит от режима работы таймера/счетчика
COM*n*B1:COM*n*B0	
COM*n*C1:COM*n*C0	
WGM*n*1:WGM*n*0	**Режим работы таймера/счетчика.** Совместно с битами WGM*n*3:WGM*n*2 регистра TCCR*n*B определяют режим работы таймера/счетчика T*n*
FOC*n*A	**Принудительное изменение состояния вывода OC*nx*.** При

FOC*n*B	записи в бит FOC*nx* лог. 1 состояние вывода OC*nx* изменяется в соответствии с установками би- тов COM*n*1*x*:COM*n*0*x* регистра TCCR*n*A. Прерывание при этом не генери- руется и сброс таймера (в режиме CTC) не производится. Эта функция до- ступна только в тех режимах, которые не используются для генерации сигнала с ШИМ. При чтении бита всегда возвращается 0

Примечание. *n* = 1, 3, 4, 5; *x* = A, B или C.

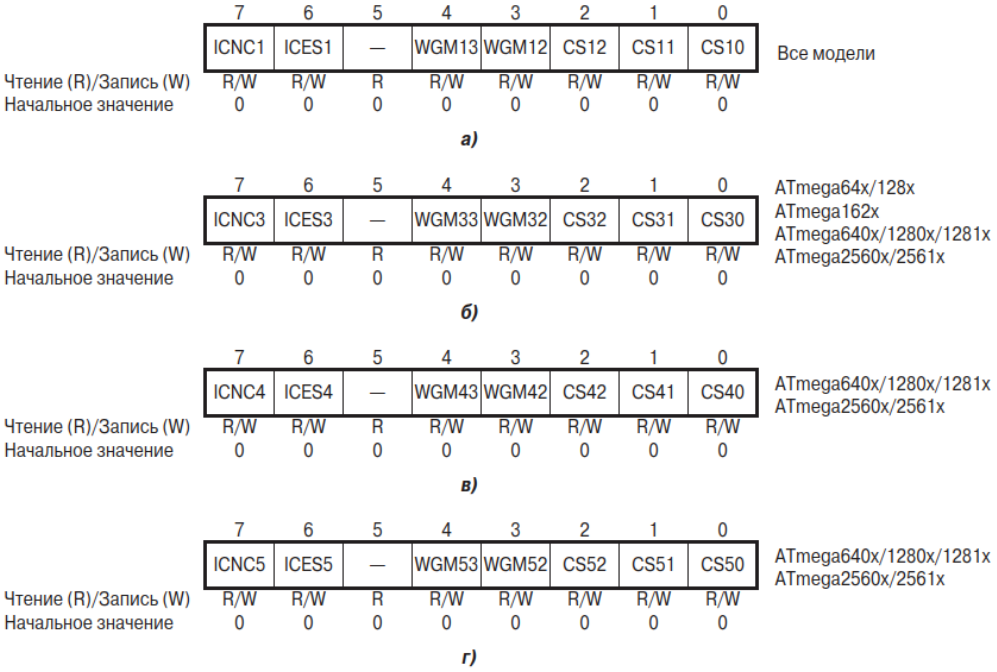


Рис.3.11. Формат регистров TCCR1B (а), TCCR3B (б), TCCR4B (в) и TCCR5B (г)

Таблица 3.7. Регистр TCCR*n*B

Бит	Название	Описание
7	ICNC*n*	**Управление схемой подавления помех блока захвата.** Если бит сброшен в 0, схема подавления помех выключена (захват производится по первому ак- тивному фронту). Если бит установлен в 1, схема подавления помех включена и захват осуществляется только в случае четырех одинаковых выборок, соответствующих активному фронту сигнала
6	ICES*n*	**Выбор активного фронта сигнала захвата.** Если бит ICES*n* сброшен в 0, со- хранение счетного регистра в регистре захвата осуществляется по спадаю- щему фронту сигнала. Если бит установлен в 1, то сохранение счетного регистра в регистре захвата осуществляется по нарастающему фронту сиг- нала. Одновременно с сохранением счетного регистра устанавливается также флаг прерывания ICF*n*

		регистра флагов
5	—	Не используется, читается как 0
4, 3	WGM*n*3:WGM*n*2	**Режим работы таймера/счетчика.** Совместно с битами WGM*n*1:WGM*n*0 регистра TCCR*n*A определяют режим работы таймера/счетчика T*n* (**Табл. 7.25**)
2…0	CS*n*2…CS*n*0	**Управление тактовым сигналом.** Эти биты определяют источник тактово- го сигнала микроконтроллера (см. подраздел 7.6.2)

Примечание. *n* = 1, 3, 4, 5.

В современном мире трудно найти область техники, где не применялись бы микропроцессоры. Они применяются при вычислениях, они выполняют

3.3. Обращение к 16-битным регистрам

Каждый 16-битный регистр таймеров/счетчиков физически размеща- ется в двух 8-битных регистрах. Соответственно, для обращения к ним требуется выполнить по две операции чтения или записи. Для того чтобы запись или чтение обоих байтов содержимого 16-битного регистра проис- ходило одновременно, в составе каждого таймера/счетчика имеется спе- циальный 8-битный регистр TEMP, предназначенный для хранения стар- шего байта значения (этот регистр используется только процессором и программно недоступен).

Для выполнения цикла записи 16-битного регистра первым должен быть загружен старший байт значения, который помещается в регистр TEMP. При последующей записи младшего байта он объединяется с со- держимым регистра TEMP, и оба байта одновременно (в одном и том же такте) записываются в 16-битный регистр. Если требуется изменить не- сколько 16-битных регистров таймера/счетчика, а старшие байты всех за- писываемых значений одинаковы, то загрузку старшего байта достаточно выполнить только один раз.

Для выполнения цикла чтения 16-битного 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. Исключение со- ставляют только регистры сравнения OCRnA/B/C, при чтении которых регистр TEMP не задействуется.

При выполнении цикла обращения к 16-битному регистру тайме- ра/счетчика прерывания должны быть запрещены. В противном случае, если прерывание произойдет между двумя командами обращения к 16-битному регистру, а в подпрограмме обработки этого прерывания тоже будет произведено обращение к какому-либо из 16-битных регистров того же таймера/счетчика, содержимое регистра TEMP будет изменено. Как следствие, результат обращения к 16-битному регистру в основной про- грамме будет

неверным.

Управление тактовым сигналом

Формирование тактового сигнала 16-битных таймеров/счетчиков clk_{Tn} (n = 1, 3, 4, 5) осуществляется блоком предделителя, который был рассмот- рен в разделе 7.4 этой главы.

В качестве тактового сигнала clk_{Tn} таймеров/счетчиков T1, T3, T4 и T5 может использоваться

- системный тактовый сигнал ($clk_{Tn} = clk_{I/O}$);
- масштабированный системный тактовый сигнал ($clk_{Tn} = clk_{I/O}/N$);
- внешний сигнал, поступающий на вход T1 (T3) микроконтроллера ($clk_{Tn} = clk_{EXT}$).

Исключение составляет лишь таймер/счетчик T3 моделей ATmega162x, который не может работать от внешнего тактового сигнала.

Выбор источника тактового сигнала, а также запуск и остановка таймеров/счетчиков осуществляются с помощью битов CS*n*2…CS*n*0 регистра управления таймером TCCR*n*B согласно **Табл.3.8.**

Таблица 3.8. **Выбор источника тактового сигнала 16-битных таймеров/счетчиков**

CS*n*2	CS*n*1	CS*n*0	Источник тактового сигнала	
			T3 в моделях ATmega162x	**Остальные**
0	0	0	Таймер/счетчик остановлен	Таймер/счетчик остановлен
0	0	1	$clk_{I/O}$	$clk_{I/O}$
0	1	0	$clk_{I/O}/8$	$clk_{I/O}/8$
0	1	1	$clk_{I/O}/64$	$clk_{I/O}/64$
1	0	0	$clk_{I/O}/256$	$clk_{I/O}/256$
1	0	1	$clk_{I/O}/1024$	$clk_{I/O}/1024$
1	1	0	$clk_{I/O}/16$	Вывод T*n*, счет осуществляется по спадающему фронту импульсов
1	1	1	$clk_{I/O}/32$	Вывод T*n*, счет осуществляется по нарастающему фронту импульсов

Примечание. n = 1, 3, 4, 5.

Режимы работы

Режим работы таймеров/счетчиков T1, T3, T4 и T5 определяется состо- янием битов WGM*n*3:WGM*n*2 регистра TCCR*n*B совместно с битами WGM*n*1:WGM*n*0 регистра TCCR*n*A. Зависимость режима работы таймеров/счетчиков от состояния этих битов показана в **Табл. 3.9.**

Таблица 3.9. **Режимы работы 16-битных таймеров/счетчиков T1, T3, T4 и T5**

Номер режима	WGM*n*3	WGM*n*2	WGM*n*1	WGM*n*0	Режим работы таймера/счетчика T*n*	Модуль счета (TOP)	Обновление регистров OCR*nx*	Момент установки флага TOV*n*
0	0	0	0	0	Normal	$FFFF	Немедленно	$FFFF
1	0	0	0	1	Phase correct PWM, 8-битный	$00FF	При TOP	$0000
2	0	0	1	0	Phase correct PWM, 9-битный	$01FF	При TOP	$0000
3	0	0	1	1	Phase correct PWM, 10-битный	$03FF	При TOP	$0000
4	0	1	0	0	CTC (сброс при совпадении)	OCR*n*A	Немедленно	$FFFF
5	0	1	0	1	Fast PWM, 8-битный	$00FF	При TOP	При TOP
6	0	1	1	0	Fast PWM, 9-битный	$01FF	При TOP	При TOP
7	0	1	1	1	Fast PWM, 10-битный	$03FF	При TOP	При TOP
8	1	0	0	0	Phase and Frequency Correct PWM	ICR*n*	$0000	$0000
9	1	0	0	1	Phase and Frequency Correct PWM	OCR*n*A	$0000	$0000
10	1	0	1	0	Phase correct PWM	ICR*n*	При TOP	$0000
11	1	0	1	1	Phase correct PWM	OCR*n*A	При TOP	$0000
12	1	1	0	0	CTC (сброс при совпадении)	ICR*n*	Немедленно	$FFFF
13	1	1	0	1	Зарезервировано	—	—	—
14	1	1	1	0	Fast PWM	ICR*n*	При TOP	При TOP
15	1	1	1	1	Fast PWM	OCR*n*A	При TOP	При TOP

Примечание. *n* = 1, 3, 4, 5.

Режим Normal

Это наиболее простой режим работы таймеров/счетчиков. В этом режиме счетный регистр функционирует как обычный суммирующий счетчик. По каждому импульсу тактового сигнала clk_{Tn} инкрементируется счетный регистр. При переходе через значение $FFFF возникает переполнение, и счет продолжается со значения $0000. В том же такте сигнала clk_{Tn}, в котором обнуляется регистр TCNT*n*, флаг прерывания по перепол- нению TOV*n* устанавливается в 1.

Блоки сравнения обоих таймеров в этом режиме могут использоваться как для генерации прерываний, так и для формирования сигналов. Состо- яние выходов OC*n*A/OC*n*B/OC*n*C каждого из блоков сравнения 16-битных таймеров/счетчиков определяется содержимым битов COM*nx*1:COM*nx*0 регистров TCCR*n*A, как показано в **Табл. 3.10.**

Таблица 3.10. **Управление выводами OC*n*A/OC*n*B/OC*n*C в режиме Normal**

COM*nx*1	COM*nx*0	Описание
0	0	Таймер/счетчик T*n* отключен от вывода OC*nx*

0	1	Состояние вывода меняется на противоположное
1	0	Вывод сбрасывается в 0
1	1	Вывод устанавливается в 1

Примечание. *n* = 1, 3, 4, 5; *x* = A, B или C.

Состояние выхода любого блока сравнения также может быть измене- но принудительно, записью лог. 1 в бит FOC*n*A/FOC*n*B/FOC*n*C регистра TCCR*n*C (в некоторых моделях — TCCR*n*A). Прерывание при этом не ге- нерируется.

Режим CTC (сброс при совпадении)

В этом режиме счетный регистр тоже функционирует как обычный сум- мирующий счетчик, инкрементирование которого осуществляется по каж- дому импульсу тактового сигнала clk_{Tn}. Однако максимально возможное значение счетного регистра и, следовательно, разрешающая способность счетчика определяются либо регистром сравнения блока A OCR*n*A (WGM*n*3:0 = 0100), либо регистром захвата ICR*n* (WGM*n*3:0 = 1100). После достижения максимального значения счет продолжается со значения $0000. Как и в режиме Normal, флаг прерывания TOV*n* устанавливается при изме- нении значения счетного регистра с $FFFF на $0000. Временные диаграммы для этого режима работы таймера/счетчика приведены на **Рис. 3.12**.

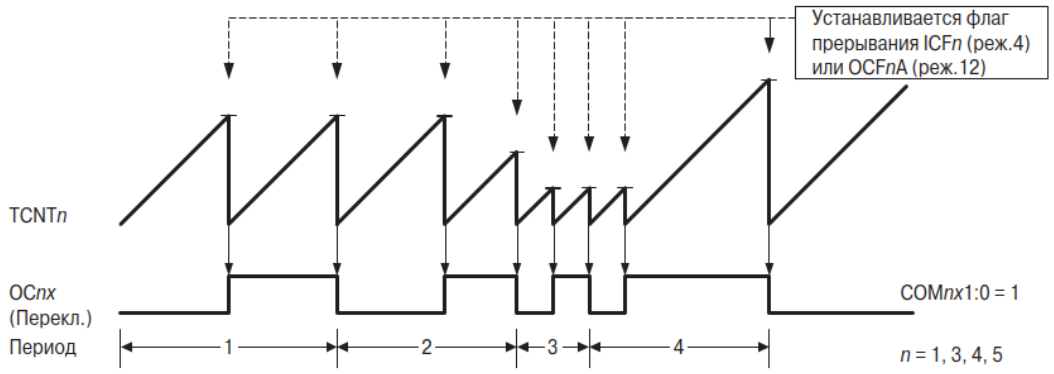


Рис.3.12. Временные диаграммы для режима CTC

При достижении счетчиком максимального значения устанавливается флаг:

- OCF*n*A, если модуль счета определяется регистром сравнения OCR*n*A (WGM*n*3:0 = 0100);
- ICF*n*, если модуль счета определяется регистром захвата ICR*n* (WGM*n*3:0 = 1100).

Одновременно с установкой флага может изменяться состояние выво- да OC*nx* микроконтроллера. Состояние вывода определяется содержимым битов COM*nx*1:COM*nx*0 регистра TCCR*nx*, как указано в **Табл. 3.11.**

Таблица 3.11. **Управление выводами OC*n*A/OC*n*B/OC*n*C в режиме CTC**

COM*nx*1	COM*nx*0	Описание
0	0	Таймер/счетчик T*n* отключен от вывода OC*nx*
0	1	Состояние вывода меняется на противоположное
1	0	Вывод сбрасывается в 0
1	1	Вывод устанавливается в 1

Примечание. *n* = 1, 3, 4, 5; *x* = A, B или C.

Для генерации сигнала заданной частоты необходимо записать в биты COM*nx*1:COM*nx*0 значение 01 (переключение состояния вывода). Частота генерируемого сигнала будет определяться выражением

$$f_{OCn} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + X)},$$

Как и в режиме Normal, состояние выводов OC*n*A/OC*n*B/OC*n*C при необходимости может быть изменено принудительно, записью лог. 1 в бит FOC*n*A/FOC*n*B/FOC*n*C регистра TCCR*n*C (в некоторых моделях — TCCR*n*A). Прерывание при этом не генерируется, и сброс счетного ре- гистра не производится.

Режим Fast PWM

Режим Fast PWM («Быстродействующий ШИМ») позволяет генериро- вать высокочастотный сигнал с широтно-импульсной модуляцией. Этот режим практически полностью идентичен одноименному режиму 8-бит- ных таймеров/счетчиков. Отличие заключается только в том, что 16-бит- ные таймеры/счетчики позволяют генерировать ШИМ-сигнал различной разрядности.

Счетный регистр в этом режиме функционирует как суммирующий счетчик, инкрементирование которого осуществляется по каждому им- пульсу тактового сигнала clkTn. Состояние счетчика изменяется от $0000 до максимального значения, после чего счетный регистр сбрасывается и цикл повторяется. В зависимости от установок битов WGMn3:0 максимальное значение счетчика (разрешение ШИМ-сигнала) либо является фиксиро- ванным значением, либо определяется содержимым определенных регист- ров таймера/счетчика (Табл. 3.12).

Таблица 3.12. Разрешающая способность модулятора в режиме Fast PWM

Номер режима	GM*n*3	GM*n*2	WGM*n*1	WGM*n*0	Разрешающая способность	Модуль счета (TOP)
5	0	1	0	1	8 битов	$00FF

6	0	1	1	0	9 битов	$01FF
7	0	1	1	1	10 битов	$03FF
14	1	1	1	0	Переменная (2…16)	ICR*n*A ($0003… $FFFF)
15	1	1	1	1	Переменная (2…16)	OCR*n*A ($0003… $FFFF)

Примечание. *n* = 1, 3, 4, 5.

При работе с какими-либо фиксированными значениями модуля счета для задания модуля рекомендуется использовать регистр захвата. При этом регистр OCRnA может использоваться для формирования ШИМ-сигнала. Если же в процессе формирования ШИМ-сигнала его частота меняется очень часто, для задания модуля счета рекомендуется использовать ре- гистр сравнения. В этом случае за счет буферизации записи в регистры сравнения исключается появление несимметричных импульсов сигнала на выходе модулятора.

При достижении счетчиком максимального значения устанавливается флаг прерывания TOVn соответствующего регистра флагов. Одновременно с ним устанавливается флаг ICFn (режим 14) либо OCFnA (режим 15).

При равенстве содержимого счетного регистра и какого-либо регистра сравнения устанавливается соответствующий флаг прерывания OCFnA/OCFnB/OCFnC. Одновременно изменяется состояние выхода блока сравнения OCnA/OCnB/OCnC. Состояние этих выходов определя ется содержимым битов COMnx1:COMnx0 регистров TCCRnA (Табл. 3.13). Временные диаграммы для случая, когда модуль счета определяется содер- жимым регистра ICRnA или OCRnA, показаны на Рис. 3.13.

Таблица 3.13. **Управление выводами OC*n*A/OC*n*B/OC*n*C в режиме Fast PWM**

COM*nx*1	**COM*nx*0**	**Описание**
0	0	Таймер/счетчик T*n* отключен от вывода OC*nx*
0	1	**OC*n*A:** WGM*n*3 = 0 — таймер/счетчик T*n* отключен от вывода OC*n*A; WGM*n*3 = 1 — состояние вывода меняется на противоположное при равенстве регистров TCNT*n* и OCR*n*A. **OC*n*B, OC*n*C:** Зарезервировано
1	0	Сбрасывается в 0 при равенстве счетного регистра и соответствующего ре- гистра сравнения. Устанавливается в 1 при достижении счетчиком макси- мального значения (неинвертированный ШИМ-сигнал)
1	1	Устанавливается в 1 при равенстве счетного регистра и соответствующего регистра сравнения. Сбрасывается в 0 при достижении счетчиком макси- мального значения (инвертированный ШИМ-сигнал)

Примечание. *n* = 1, 3, 4, 5; *x* = A, B или C.

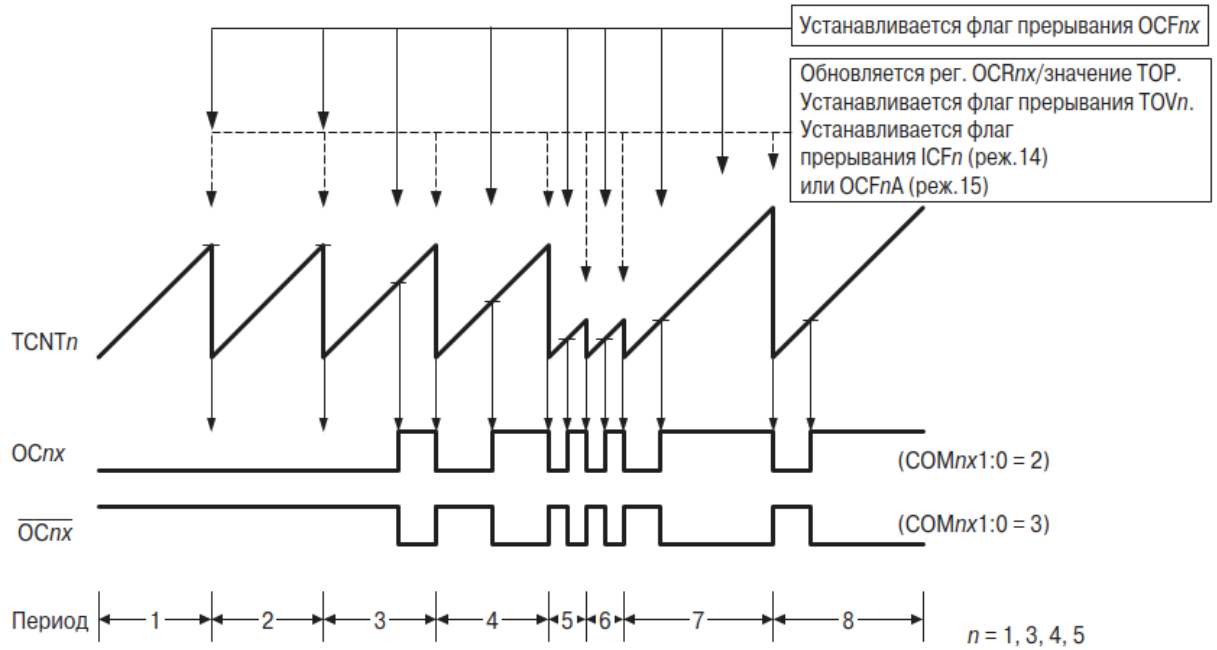


Рис.3.13. Формирование ШИМ-сигнала в режиме Fast PWM

Если содержимое регистра сравнения равно модулю счета, то выход соответствующего блока сравнения переключится в устойчивое состоя- ние, определяемое установками битов COMnx1:COMnx0 (Рис. 3.13).

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, состояние которого сначала изменяется от $0000 до максимального значения, а затем обратно до $0000. Соответственно, максимальная частота сигнала в этом режиме в 2 раза ниже максимальной частоты сигнала в режиме Fast PWM.

В зависимости от установок битов WGMn3:0 максимальное значение счетчика (разрешение ШИМ-сигнала) либо является фиксированным значением, либо определяется содержимым определенных регистров таймера/счетчика (Табл. 3.14).

Таблица 3.14. **Разрешающая способность модулятора в режиме Phase Correct PWM**

Номер режима	WGM*n*3	WGM*n*2	WGM*n*1	WGM*n*0	Разрешающая способность	Модуль счета (TOP)
1	0	0	0	1	8 битов	$00FF
2	0	0	1	0	9 битов	$01FF
3	0	0	1	1	10 битов	$03FF

10	1	0	1	0	Переменная (2…16)	ICR*n*A ($0003…$FFFF)
11	1	0	1	1	Переменная (2…16)	OCR*n*A ($0003…$FFFF)

Примечание. *n* = 1, 3, 4, 5.

Как и в режиме Fast PWM, при работе с какими-либо фиксированны- ми значениями модуля счета для задания модуля рекомендуется использо- вать регистр захвата. При этом регистр OCR*n*A может использоваться для формирования ШИМ-сигнала. Если же в процессе формирования ШИМ-сигнала его частота меняется очень часто, для задания модуля счета рекомендуется использовать регистр сравнения.

При достижении счетчиком максимального значения происходит смена направления счета, но счетчик остается в этом состоянии в течение одного периода сигнала clk_{Tn}. В этом же такте производится обновление содержимого регистра сравнения. Если модуль счета определяется регист- ром сравнения ICR*n*A (режим 10) или OCR*n*A (режим 11), одновременно с обновлением регистра сравнения устанавливается флаг ICF*n* либо OCF*n*A соответственно.

При достижении счетчиком минимального значения ($0000) также происходит смена направления счета и одновременно устанавливается флаг прерывания TOV*n* соответствующего регистра флагов. При равенстве содержимого счетного регистра и какого-либо регистра сравнения уста- навливается соответствующий флаг OCF*n*A/OCF*n*B/OCF*n*C. Одновре- менно изменяется состояние выхода блока сравнения OC*n*A/OC*n*B/OC*n*C. Как обычно, состояние вывода определяется содержимым битов COM*nx*1:COM*nx*0 регистров TCCR*n*A (**Табл. 3.15**). Временные диаграммы для случая, когда модуль счета определяется содержимым регистра ICR*n*A или OCR*n*A, показаны на **Рис. 3.14**.

Таблица 3.15. **Управление выводами OC*n*A/OC*n*B/OC*n*C в режиме Phase Correct PWM**

COM*nx*1	**COM*nx*0**	**Описание**
0	0	Таймер/счетчик T*n* отключен от вывода OC*nx*
0	1	**OC*n*A:** WGM*n*3 = 0 — таймер/счетчик T*n* отключен от вывода OC*n*A; WGM*n*3 = 1 — состояние вывода меняется на противоположное при ра- венстве регистров TCNT*n* и OCR*n*A. **OC*n*B, OC*n*C:** Зарезервировано
1	0	Сбрасывается в 0 при прямом счете и

		устанавливается в 1 при обратном счете (неинвертированный ШИМ-сигнал)
1	1	Устанавливается в 1 при прямом счете и сбрасывается в 0 при обратном счете (инвертированный ШИМ-сигнал)

Примечание. *n* = 1, 3, 4, 5; *x* = A, B или C.

Следует понимать, что при изменении модуля счета во время работы таймера/счетчика на выходе блоков сравнения могут появиться несиммет ричные (относительно середины периода модуляции) импульсы. Посколь- ку обновление содержимого регистра сравнения происходит в момент до- стижения счетчиком максимального значения, период ШИМ-сигнала ра- вен времени между этими моментами. При этом время обратного счета определяется предыдущим значением модуля счета, а время прямого счета — новым значением. Если эти значения различны, то время прямого и время обратного счета также отличаются. Результатом этого и являются несимметричные импульсы на выходе блоков сравнения, как показано на **Рис. 3.14** (3-й период сигнала).

Поэтому при частом изменении модуля счета во время работы тайме- ра/счетчика рекомендуется использовать режим Phase and Frequency Correct PWM, описанный в следующем подразделе. Если же используется постоянное значение модуля счета, то между этими двумя режимами нет никакой разницы.

Если значение, находящееся в регистре сравнения, равно $0000 или модулю счета (*TOP*), то при следующем совпадении состояния счетчика и содержимого регистра сравнения выход схемы сравнения переключится в устойчивое состояние согласно **Табл. 3.16.**

Таблица 3.16. **Устойчивые состояния выхода схемы сравнения**

COM*nx*1	**COM*nx*0**	**Регистр OCR*nx***	**Состояние вывода OC*nx***
1	0	$0000	0
1	0	TOP	1
1	1	$0000	1
1	1	TOP	0

Примечание. *n* = 1, 3, 4, 5; *x* = A, B или C.

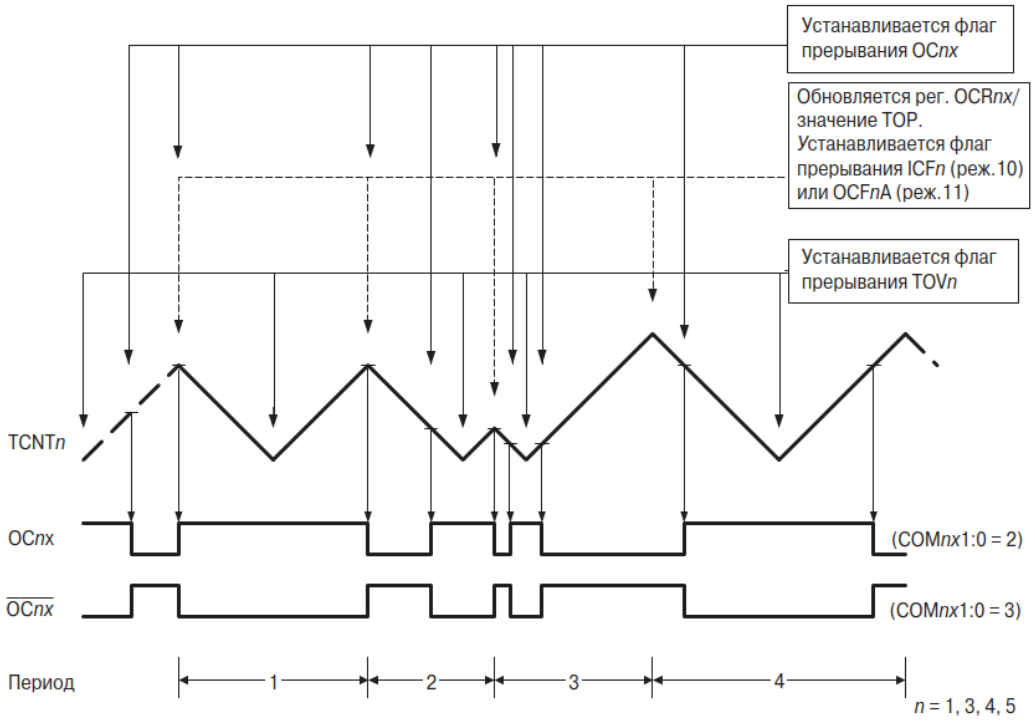


Рис.3.14. Формирование ШИМ-сигнала в режиме Phase Correct PWM

Режим Phase and Frequency Correct PWM

Режим Phase and Frequency Correct PWM («ШИМ с точной фазой и частотой») очень похож на режим Phase Correct PWM. Единственная принципиальная разница между ними — момент обновления содержимого регистра сравнения.

Максимальное значение счетчика (разрешение ШИМ-сигнала) в этом режиме может определяться только регистрами ICRnA или OCRnA таймера/счетчика, как показано в Табл.3.17.

Таблица 3.17. **Разрешающая способность модулятора в режиме Phase and Frequency Correct PWM**

Номер режима	***WGMn3***	***WGMn2***	**WGMn1**	**WGMn0**	**Разрешающая способность**	**Модуль счета (TOP)**
8	1	0	1	0	Переменная (2…16)	ICRnA ($0003…$FFFF)
9	1	0	1	1	Переменная (2…16)	OCRnA ($0003…$FFFF)

Примечание. *n* = 1, 3, 4, 5

Как и в остальных режимах, при работе с какими-либо фиксированны- ми значениями модуля счета для задания модуля рекомендуется использо- вать регистр захвата. При этом регистр OCR*n*A может использоваться для формирования ШИМ-сигнала. Если же в процессе формирования ШИМ-

сигнала его частота меняется очень часто, то для задания модуля счета рекомендуется использовать регистр сравнения.

При достижении счетчиком максимального значения происходит смена направления счета, но счетчик остается в этом состоянии в течение одного периода сигнала clk_{Tn}. В этом же такте устанавливается флаг ICF*n* либо OCF*n*A (в зависимости от того, какой из регистров используется для задания модуля счета).

При достижении счетчиком минимального значения ($0000) направление счета опять изменяется. При этом устанавливается флаг прерывания TOV*n* и производится обновление содержимого регистра сравнения.

При равенстве содержимого счетного регистра и какого-либо регистра сравнения устанавливается соответствующий флаг OCF*n*A/OCF*n*B/OCF*n*C соответствующего регистра. Одновременно изменяется состояние выхода блока сравнения OC*n*A/OC*n*B/OC*n*C. Как обычно, состояние вывода опреде- ляется содержимым битов COM*nx*1:COM*nx*0 регистров TCCR*n*A (**Табл. 3.17**и **Рис. 3.15**).

Таблица 3.17. **Управление выводами OC*n*A/OC1B/OC*n*C в режиме Phase and Frequency Correct PWM**

COM*nx*1	**COM*nx*0**	**Описание**
0	0	Таймер/счетчик T*n* отключен от вывода OC*n*A/OC*n*B/OC*n*C
0	1	**OC*n*A:** WGM*n*3 = 0 — таймер/счетчик T*n* отключен от вывода OC*n*A; WGM*n*3 = 1 — состояние вывода меняется на противоположное при ра- венстве регистров TCNT*n* и OCR*n*A. **OC*n*B, OC*n*C:** Зарезервировано
1	0	Сбрасывается в 0 при прямом счете и устанавливается в 1 при обратном счете (неинвертированный ШИМ-сигнал)
1	1	Устанавливается в 1 при прямом счете и сбрасывается в 0 при обратном счете (инвертированный ШИМ-сигнал)

Примечание. *n* = 1, 3, 4, 51; *x* = A, B или C.

Если сравнить **Рис. 3.14** и **3.15**, можно увидеть, что в режиме Phase and Frequency Correct PWM каждый период сигнала является полностью симметричным. Это следствие того, что обновление содержимого регистра сравнения происходит в момент достижения счетчиком минимального значения. Поэтому время прямого счета всегда равно времени обратного счета, выходные импульсы симметричны, и соответственно частота гене- рируемого сигнала остается постоянной.

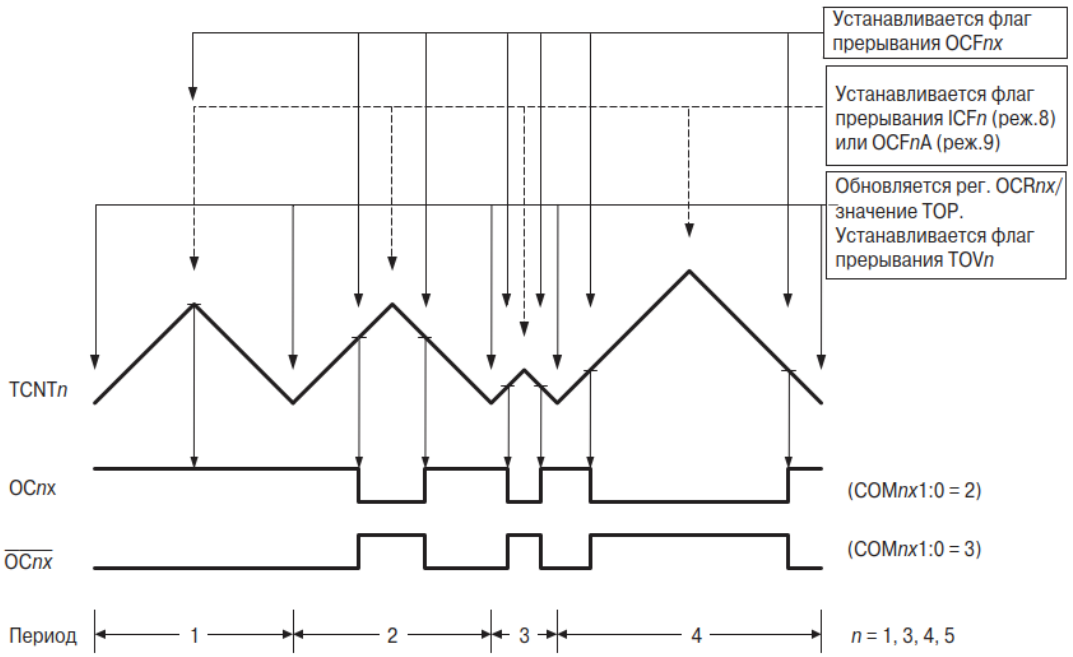


Рис.3.15. Формирование ШИМ-сигнала в режиме Phase and Frequency Correct PWM

Формирование ШИМ-сигнала в режиме Phase and Frequency Correct PWM

Если значение, находящееся в регистре сравнения, равно $0000 или модулю счета (TOP), то при следующем совпадении состояния счетчика и содержимого регистра сравнения выход схемы сравнения переключится в устойчивое состояние согласно **Табл. 3.18**.

***Таблица 3.18.* Устойчивые состояния выхода схемы сравнения**

COM*nx*1	**COM*nx*0**	**Регистр OCR*nx***	**Состояние вывода OC*nx***
1	0	$0000	0
1	0	TOP	1
1	1	$0000	1
1	1	TOP	0

Примечание. *n* = 1, 3, 4, 5; *x* = A, B или C.

Вопросы по 3 главе:

1. Какие 16-битные таймеры/счётчики присутствуют в ATmega48x/88x/168x и сколько у них каналов сравнения (OCRxA, OCRxB, OCRxC)?
2. Какие режимы работы 16-битного таймера позволяют формировать ШИМ с переменной разрешающей способностью (TOP, задаваемый ICR1 или OCR1A)?
3. В чём отличие режима Phase Correct PWM от режима Fast PWM для 16-битного таймера? Как это влияет на максимальную частоту выходного сигнала?
4. Для чего предназначен регистр захвата ICR1? Как настраиваются активный фронт (ICES1) и подавление помех (ICNC1)?
5. Что такое асинхронный режим работы таймера T2 (бит AS2 в регистре ASSR) и для каких целей он обычно используется?
6. Какие флаги обновления (TCN2UB, OCR2AUB, OCR2BUB) контролируют готовность регистров при работе таймера T2 в асинхронном режиме?
7. Какие биты регистра TCCR1B управляют источником тактового сигнала для 16-битного таймера T1? Что означает комбинация CS12…CS10 = 110?

Глава 4. Сторожевой таймер.Модулятор

4.1. Общие сведения

В микроконтроллерах ATmega640x/1280x/1281x/2560x/2561x появился новый блок, отсутствующий в других моделях, — модулятор (Output Compare Modulator). С его помощью можно формировать сигналы, моду- лированные несущей частотой. Для генерации сигнала несущей частоты используется блок сравнения A таймера/счетчика T0, а для формирования модулирующего сигнала — блок сравнения C таймера/счетчика T1. Структурная схема модулятора показана на **Рис. 4.1.**

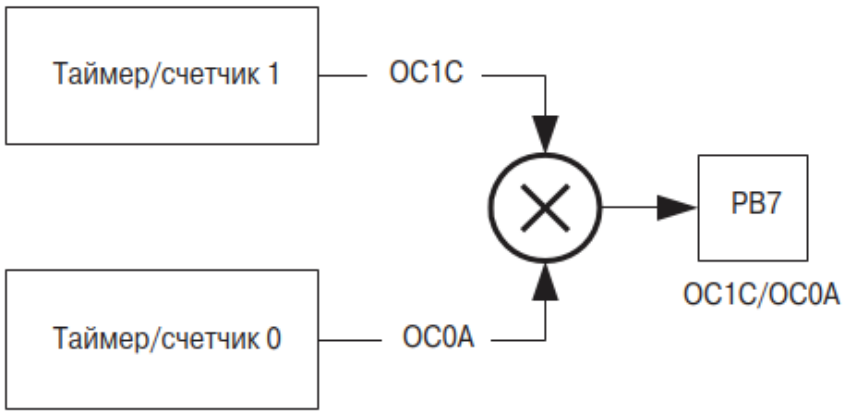


Рис.4.1. Структурная схема модулятор

Выходы упомянутых блоков сравнения (OC1C и OC0A) задействуют один и тот же вывод микроконтроллера — 7-й бит порта B. Поэтому модулятор включается автоматически при одновременном разрешении работы этих блоков сравнения (т. е. при значении битов COMnx1:0, отличном от 0). Тип модуляции («Логическое И» или «Логическое ИЛИ») определяется битом PORTB7 регистра PORTB.Упрощенная схема модулятора приведена на Рис. 4.2, а временные диаграммы работы модулятора — на Рис. 4.3.

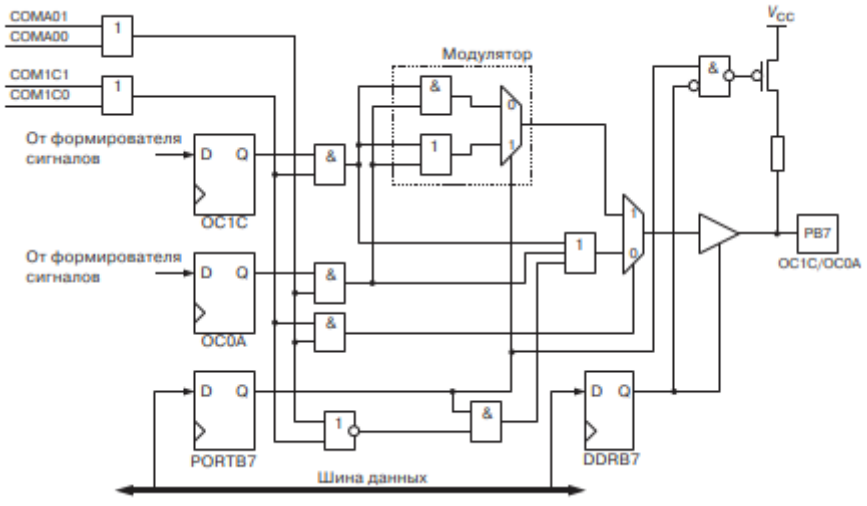


Рис.4.2. Упрощенная схема модулятора

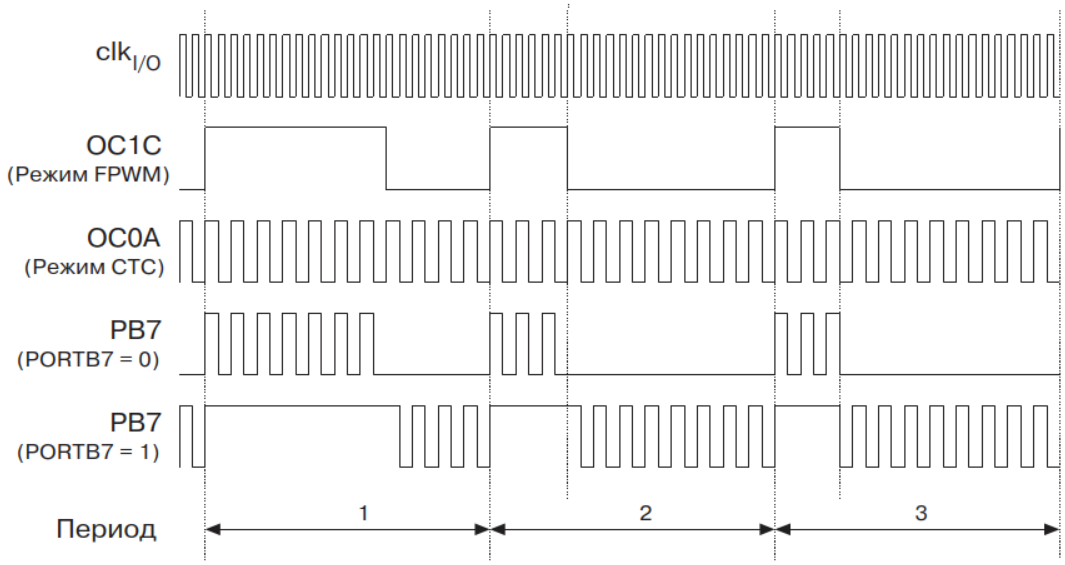


Рис.4.3. Временные диаграммы работы модулятора

Обратите внимание, что в результате модуляции снижается разреше- ние ШИМ-сигнала (OC1C). Коэффициент уменьшения равен количеству тактов системного тактового сигнала, укладывающихся в один период сигнала несущей частоты (OC0A). В случае, показанном на Рис. 4.3, разрешающая способность уменьшается в 2 раза. Причину такого снижения разрешающей способности можно понять, посмотрев на выходной сигнал модулятора (2-й и 3-й периоды при PORTB7 = 0). Несмотря на то что длительность ВЫСОКОГО уровня модулирующего сигнала во 2-м периоде на один такт больше, чем в 3-м периоде, итоговый сигнал в обоих периодах одинаков.

4.2. Сторожевой таймер

Все микроконтроллеры семейства Mega имеют в своем составе сторожевой таймер, предназначенный для защиты микроконтроллера от сбоев в процессе работы. В общей сложности в микроконтроллерах семейства можно встретить сторожевой таймер двух исполнений — стандартный и расширенный. Во всех старых моделях и в некоторых новых реализован стандартный сторожевой таймер, структурная схема которого приведена на Рис. 4.4, а. В большинстве же новых микроконтроллеров реализован так называемый расширенный сторожевой таймер, структурная схема которого приведена на Рис. 4.4, б. Основным отличием нового сторожевого таймера является возможность генерации прерывания по его тайм-ауту.

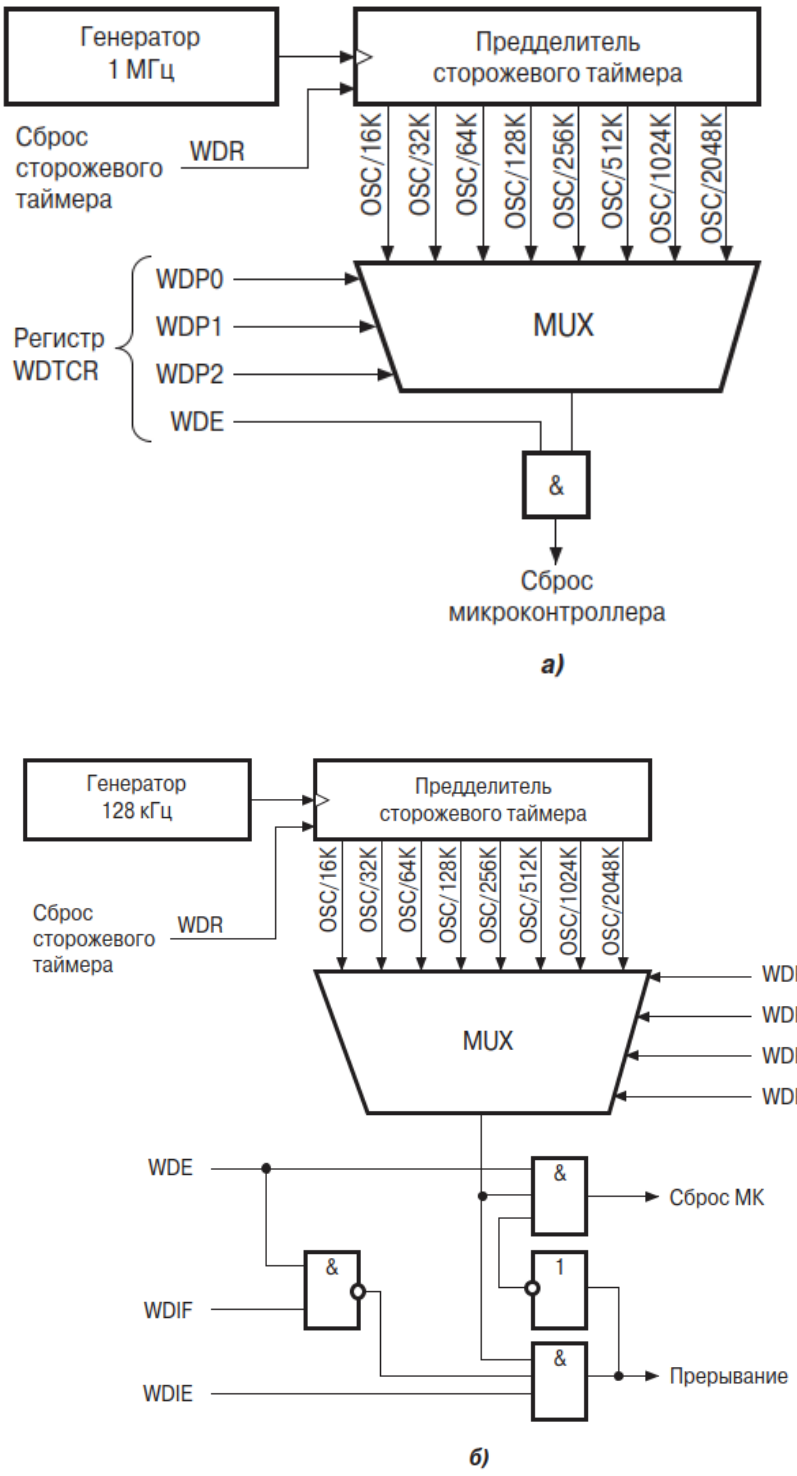


Рис.4.4 Структурная схема сторожевого таймера:
а - стандартный сторожевой таймер;
б - расширенный сторожевой таймер.

Сторожевой таймер обоих исполнений имеет независимый тактовый генератор, поэтому он работает даже во время нахождения микроконтроллера в любом из спящих режимов. Типовое значение частоты этого генератора равно 1 МГц (стандартный таймер) или 128 кГц (расширенный тай мер) при V_{CC} = 5.0 В. Фактическая частота генератора зависит от напряжения питания устройства, температуры, технологического разброса.

Если сторожевой таймер включен, то через промежутки времени, рав- ные его периоду, он выполняет сброс микроконтроллера (как уже упоми-

налось, расширенный сторожевой таймер может также генерировать прерывание). Чтобы избежать сброса при нормальном выполнении программы, сторожевой таймер необходимо регулярно сбрасывать через промежутки времени, меньшие его периода. Сброс сторожевого таймера осуществляется командой WDR.

Конфигурирование сторожевого таймера в различных моделях осуществляется по-разному.

ATmega16x/32x

В этих моделях для управления сторожевым таймером предназначен регистр WDTCR, расположенный по адресу $21 ($41). Формат этого регистра приведен на Рис.4.5, а описание его битов — в Табл. 4.1.

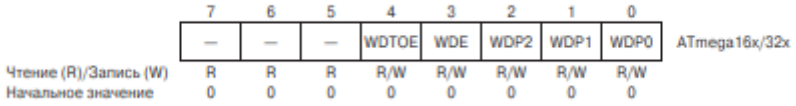


Рис.4.5. Формат регистра WDTCR

Таблица 4.1. **Биты регистра WDTCR**

Бит	Название	Краткое описание
7…5	—	Зарезервировано, читается как 0
4	WDTOE	Разрешение выключения сторожевого таймера
3	WDE	Разрешение сторожевого таймера (1 — включен)
2	WDP2	Коэффициент деления предделителя сторожевого таймера
1	WDP1	
0	WDP0	

Для включения/выключения сторожевого таймера используются два бита регистра WDTCR — WDE и WDTOE. Если бит WDE установлен в 1, сторожевой таймер включен, если сброшен в 0 — выключен. Причем для сброса бита WDE необходимо выполнить следующие действия:

1. Одной командой записать лог. 1 в биты WDE и WDTOE.
2. В течение следующих четырех тактов записать лог. 0 в бит WDE. Период тайм-аута сторожевого таймера задается с помощью битов WDP2…WDP0 регистра WDTCR согласно **Табл. 4.2.**

WDP2	WDP1	WDP0	Число тактов генератора	Период тайм-аута (типовое значение) [мс]	
				V_{CC} = 3.0 В	V_{CC} = 5.0 В
0	0	0	16K (16384)	17.3	16.3
0	0	1	32K (32768)	34.3	32.5
0	1	0	64K (65536)	68.5	65
0	1	1	128K (131072)	140	130
1	0	0	256K (262144)	270	260
1	0	1	512K (524288)	550	520
1	1	0	1024K (1048576)	1100	1000
1	1	1	2048K (2097152)	2200	2100

Табл.4.2. Задание периода сторожевого таймера

Чтобы избежать непреднамеренного сброса микроконтроллера при изменении периода сторожевого таймера, необходимо перед записью битов WDP2:0 либо запретить сторожевой таймер, либо сбросить его.

ATmega8515x/8535x, ATmega8x, ATmega64x/128x, ATmega162x, ATmega165, ATmega325x/3250x/645x/6450x

В этих моделях для управления сторожевым таймером также использу-ется регистр WDTCR, расположенный по адресу $21 ($41). Формат этого регистра приведен на **Рис. 4.6**, а описание его битов — в **Табл. 4.3.**

	7	6	5	4	3	2	1	0	
	—	—	—	WDCE	WDE	WDP2	WDP1	WDP0	ATmega8515x/8535x ATmega8x/64x/128x ATmega162x ATmega165x ATmega325x/3250x ATmega645x/6450x
Чтение (R)/Запись (W)	R	R	R	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Рис.4.6. Формат регистра WDTCR

Бит	Название	Краткое описание
7...5	—	Зарезервировано, читается как 0
4	WDCE	Разрешение изменения конфигурации сторожевого таймера
3	WDE	Разрешение сторожевого таймера (1 — включен)
2	WDP2	Коэффициент деления предделителя сторожевого таймера
1	WDP1	
0	WDP0	

Табл.4.3. Биты регистра WDTCR

Для включения/выключения сторожевого таймера используются два бита регистра WDTCR — WDE и WDTOE. Если бит WDE установлен в 1, сторожевой таймер включен, если сброшен в 0 — выключен.

В этих моделях предусмотрено несколько так называемых уровней безопасности, каждый из которых накладывает определенные огра- ничения на

изменение конфигурации сторожевого таймера. Для выбора конкретного уровня почти во всех указанных моделях используются две конфигурационные ячейки (в моделях ATmega8x, ATmega165x и ATmega325x/3250x/645x/6450x — одна). Первая ячейка, общая для всех мо- делей, называется WDTON. Второй ячейкой, определяющей уровень безо- пасности, служит ячейка, которая переводит микроконтроллер в режим совместимости с какими-либо другими моделями. Соответственно, в мо- делях ATmega8515x это ячейка S8515C, в моделях ATmega8535x — ячейка S8535C, в моделях ATmega162x — M161C, а в моделях ATmega64x/128x — M103C. Соответствие между состоянием этих ячеек и конфигурацией сторожевого таймера показано в Табл.4.4.

Таблица 4.4. **Конфигурация сторожевого таймера**

S8515C (ATmega8515x), S8535C (ATmega8535x), M161C (ATmega162x), M103C (ATmega64x/128x)	**WDTON**	**Уровень**	**Начальное состояние сторожевого таймера**	**Выключение сторожевого таймера**	**Изменение периода тайм-аута**
1	1	1	Выключен	Последователь- ность команд	Последователь- ность команд
1	0	2	Включен	Всегда включен	Последовательность команд
0	1	0[1)]	Выключен	Последователь- ность команд	Без ограничений
0	0	2[1)]	Включен	Всегда включен	Последовательность команд
[1)]Отсутствует в моделях ATmega8x, ATmega165x и ATmega325x/3250x/645x/6450x.					

Режим управления в рассматриваемых моделях характеризуется тремя уровнями: 0, 1 и 2.

Уровень 0. В этом режиме управление сторожевым таймером осуществ- ляется так же, как в более ранних моделях (см. описание предыдущей группы микроконтроллеров). При включении микроконтроллера сторо- жевой таймер выключен, однако он может быть включен в любой момент записью лог. 1 в бит WDE регистра WDTCR.

Уровень 1. Для выключения сторожевого таймера или для изменения периода тайм-аута необходимо выполнить следующие действия:

1. Одной командой записать лог. 1 в биты WDE и WDTCE.
2. В течение следующих четырех тактов записать (одной командой) требуемые значения в биты WDE и WDP2:0, одновременно сбрасы- вая бит WDCE.

Уровень 2. В этом режиме сторожевой таймер включен постоянно (бит WDE всегда читается как 1) и не может быть выключен. Для изменения пе- риода тайм-аута необходимо выполнить следующие действия:

1. Одной командой записать лог. 1 в биты WDE и WDTCE.
2. В течение следующих четырех тактов записать требуемое значение в биты WDP2:0, одновременно сбрасывая бит WDCE. Значение, запи- сываемое в

бит WDE, безразлично.

Период тайм-аута сторожевого таймера задается с помощью битов WDP2…WDP0 регистра WDTCR согласно **Табл. 4.5.**

WDP2	WDP1	WDP0	Число тактов генератора	Период тайм-аута (типовое значение) [мс]	
				V_{CC} = 3.0 В	V_{CC} = 5.0 В
0	0	0	16K (16384)	17.3	16.3
0	0	1	32K (32768)	34.3	32.5
0	1	0	64K (65536)	68.5	65
0	1	1	128K (131072)	140	130
1	0	0	256K (262144)	270	260
1	0	1	512K (524288)	550	520
1	1	0	1024K (1048576)	1100	1000
1	1	1	2048K (2097152)	2200	2100

Табл.4.5. Задание периода сторожевого таймер

Как и прежде, перед записью битов WDP2:0 рекомендуется либо запре- тить сторожевой таймер, либо сбросить его.

ATmega48x/88x/168x, ATmega164x/324x/644x, ATmega640x/1280x/1281x/2560x/2561x

В этих моделях реализован расширенный сторожевой таймер, основ- ным отличием которого от обычного таймера является возможность рабо- ты в нескольких режимах. Частота генератора расширенного сторожевого таймера была уменьшена до 128 кГц, благодаря чему максимальный пери- од тайм-аута увеличился до 8 с.

Для управления расширенным сторожевым таймером используется ре- гистр управления WDTCSR, расположенный по адресу ($60). Формат это- го регистра приведен на Рис. 4.7, а описание его битов — в Табл. 4.6.

	7	6	5	4	3	2	1	0	
	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	ATmega48x/88x/168x ATmega164x/324x/644x ATmega640x/1280x/1281x ATmega2560x/2561x
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	

Рис.4.7. Формат регистра WDTCSR

Бит	Название	Краткое описание
7	WDIF	Флаг прерывания от сторожевого таймера
6	WDIE	Разрешение прерывания от сторожевого таймера
5	WDP3	Коэффициент деления предделителя сторожевого таймера (совместно с битами WDP2...WDP0)
4	WDCE	Разрешение изменения конфигурации сторожевого таймера
3	WDE	Разрешение сброса по тайм-ауту сторожевого таймера
2	WDP2	Коэффициент деления предделителя сторожевого таймера
1	WDP1	
0	WDP0	

Табл.4.6. Биты регистра WDTCSR

Для включения/выключения сторожевого таймера используются два бита регистра WDTCR — WDE и WDCE. Если бит WDE установлен в 1, сторожевой таймер включен, если сброшен в 0 — выключен. Для сброса бита WDE необходимо выполнить следующие действия:

1. Одной командой записать лог. 1 в биты WDE и WDCE.
2. В течение следующих четырех тактов записать лог. 0 в бит WDE. Обратите внимание на то, что состояние флага сброса WDRF регистра

MCUSR отменяет состояние бита WDE. Это означает, что бит WDE установлен всегда, когда установлен флаг WDRF, поэтому перед сбросом WDE необходимо также сбросить WDRF.

Разрешение прерывания от сторожевого таймера осуществляется установкой в 1 бита WDIE регистра WDTCSR. Для индикации прерывания служит флаг WDIF того же регистра. При генерации запроса на прерыва- ние этот флаг устанавливается в 1. Сбрасывается он аппаратно при запуске соответствующей подпрограммы обработки прерывания или же вручную, записью в него лог. 1.

Как уже упоминалось, расширенный сторожевой таймер может работать в нескольких режимах (Табл.4.7). В режиме сброса он работает как стан- дартный сторожевой таймер, вызывая сброс микроконтроллера через задан- ные промежутки времени. Как правило, этот режим используется для предотвращения зависания программы из-за сбоя в работе микроконтроллера.

В режиме прерывания по тайм-ауту сторожевого таймера вместо сбро- са микроконтроллера генерируется прерывание. Это прерывание можно использовать для вывода микроконтроллера из «спящего» режима или же в качестве системного таймера, например для ограничения времени выпол- нения какой-либо операции. Режим прерывания включается установкой бита WDIE при сброшенном бите WDE.

Третий режим — режим прерывания и сброса — объединяет в себе два предыдущих. В этом режиме по первому тайм-ауту сторожевого таймера генерируется прерывание, после завершения обработки которого стороже- вой таймер автоматически переключается в режим сброса. Соответственно при последующем тайм-ауте выполняется сброс микроконтроллера. Дан- ный режим позволяет перед сбросом микроконтроллера сохранить раз- личные критические переменные программы.

Режим сброса и прерывания включается одновременной установкой

битов WDE и WDIE. При выполнении подпрограммы обработки прерыва- ния флаги WDIE и WDIF автоматически сбрасываются (таймер переклю- чается в режим сброса). Чтобы сторожевой таймер остался в режиме сбро- са и прерывания, бит WDIE необходимо устанавливать после каждого пре- рывания, причем это рекомендуется осуществлять вне обработчика прерывания.

Таблица 4.7. **Режимы работы сторожевого таймера**

WDTON	**WDE**	**WDIE**	**Режим**	**Действие по тайм-ауту**
1	0	0	Остановлен	Нет действий
1	0	1	Режим прерывания	Прерывание
1	1	0	Режим сброса	Сброс
1	1	1	Режим прерывания и сброса	Прерывание, затем переключение в режим сброса
0	X	X	Режим сброса	Сброс

Период тайм-аута сторожевого таймера задается с помощью битов WDP3…WDP0 регистра WDTCSR согласно **Табл. 4.8.**

Для изменения периода тайм-аута необходимо выполнить следующие действия:

1. Одной командой записать лог. 1 в биты WDE и WDCE.
2. В течение следующих четырех тактов записать требуемое значение в биты WDP3:0 и WDE, одновременно сбрасывая бит WDCE.

Перед изменением битов WDP3:0 сторожевой таймер рекомендуется сбрасывать.

WDP3	WDP2	WDP1	WDP0	Число тактов генератора	Период тайм-аута (типовое значение) при V_{CC} = 5.0 В
0	0	0	0	2K (2048)	16 мс
0	0	0	1	4K (4096)	32 мс
0	0	1	0	8K (8192)	64 мс
0	0	1	1	16K (16384)	0.125 с
0	1	0	0	32K (32768)	0.25 с
0	1	0	1	64K (65536)	0.5 с
0	1	1	0	128K (131072)	1.0 с
0	1	1	1	256K (262144)	2.0 с
1	0	0	0	512K (524288)	4.0 с
1	0	0	1	1024K (1048576)	8.0 с
1	0	1	0	Зарезервировано	
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

Табл.4.8. Задание периода сторожевого таймера

Вопросы по 4 главе:

1. Для чего предназначен сторожевой таймер (Watchdog Timer) в микроконтроллерах и как он помогает предотвратить зависание программы?
2. В чём отличие расширенного сторожевого таймера (с регистром WDTCSR) от стандартного (с регистром WDTCR)?
3. Как изменить период тайм-аута сторожевого таймера? Опишите последовательность записи в биты WDCE и WDE.
4. Что такое модулятор (Output Compare Modulator) в моделях ATmega640x/1280x и как он формирует модулированный сигнал?
5. Какие таймеры/счётчики используются в модуляторе для несущей (OC0A) и модулирующей (OC1C) частоты?
6. Какой вывод микроконтроллера является выходом модулятора и как тип модуляции (AND или OR) задаётся через PORTB7?
7. Почему при использовании модулятора снижается разрешающая способность ШИМ-сигнала OC1C?
8. Какое типовое значение частоты генератора расширенного сторожевого таймера и какой максимальный период тайм-аута можно получить?

Список литературы

1. Основы микропроцессорной техники / Ю.В. Новиков, П.К. Скоробогатов. – М.: Интернет – Университет Информационных Технологий, 2003. – 440 с.
2. Микропроцессорные системы: Учеб. Пособие для вузов / Е.К. Александров, Р.И. Грушвицкий, М.С. Куприянов и др.; под общ. Ред. Д.В. Пузанкова. – СПб.: Политехника, 2002. – 935 с.: ил.
3. Современные микроконтроллеры и микропроцессоры Motorola: Справочник. – М.: Горячая линия – Телеком, 2004. – 952 с.: ил.
4. Работа с микроконтроллерами семейства HC(S)08: Учеб. Пособие для студентов технических вузов / Х. Крейдл, Г. Куприс, Т.В. Ремизевич и др.; под ред. Д.И. Панфилова. – М.: Изд-во МЭИ, 2005. – 444 с.: ил.
5. Микроконтроллеры. Разработка встраиваемых приложений / А.Е. Васильев – СПб.: БХВ – Петербург, 2008. – 304 с.: ил.
6. Архитектура микропроцессорных систем / Б.В. Костров, В.Н. Ручкин. – М.: Изд-во Диалог–МИФИ, 2007. – 304 с.: табл. 14, ил. 147.
7. Создаем устройства на микроконтроллерах / А.В. Белов– СПб.: Наука и Техника, 2007. – 304 с.: ил.

Дополнительная:

8. Микроконтроллеры для встраиваемых приложений: от общих подходов – к семействам HC05 и HC08 фирмы Motorola / Под ред. И.С. Кирюхина. – М.: ДОДЭКА, 2000. – 272 с.
9. Руководство по микроконтроллерам / М. Предко. – Т.1. – М.: Постмаркет, 2001. – 411 с.
10. Руководство по микроконтроллерам / М. Предко. – Т.1. – М.: Постмаркет, 2001. — 488 с.
11. Самоучитель по микропроцессорной технике / А.В. Белов. – СПб.: Наука и техника, 2003. – 224 с.
12. Микропроцессорные системы бытовой техники / Б.П. Баев – М.: Легкая промышленность и бытовое обслуживание, 2001. – 464 с.
13. Цифровые устройства и микропроцессорные системы: Учебник для техникумов связи / Б.А. Калабеков. – М.: Горячая линия – Телеком, 2000. – 336 с.: ил.

Приложение 1

Задания на курсовой проект

Спроектировать управляющее устройство на базе микроконтроллера Atmega328 семейства AVR. Разработать, описать функциональную и принципиальную электрическую схемы устройства (формат А3). Разработать алгоритм функционирования устройства, блок-схему и программу на языке ассемблера микроконтроллера Atmega328. Описать процесс отладки управляющей программы в интегрированной среде разработки.

1. Разработать генератор прямоугольных импульсов. Частота импульсов fвых на выходе генератора в герцах 1, 10 задается переключателями и в двоично-десятичном коде отображается на цифровом индикаторе. Длительность импульса 500мс, 50мс соответственно. Кнопка ПУСК запускает генератор. Кнопка СТОП – выключает.
2. Разработать генератор прямоугольных импульсов. Частота импульсов fвых на выходе генератора в герцах от 5, 25 задается переключателями в двоично-десятичном коде и отображается на цифровом индикаторе. Длительность импульса 100мс, 20мс соответственно. Кнопка ПУСК запускает генератор. Кнопка СТОП – выключает.
3. Разработать генератор прямоугольных импульсов. Частота импульсов fвых на выходе генератора в герцах 10, 100 задается переключателями и в двоично-десятичном коде отображается на цифровом индикаторе. Длительность импульса 50мс и 5мс соответственно. Кнопка ПУСК запускает генератор. Кнопка СТОП – выключает.
4. Разработать генератор прямоугольных импульсов. Частота импульсов fвых на выходе генератора в герцах от 25, 50 задается переключателями в двоично-десятичном коде и отображается на цифровом индикаторе. Длительность импульса 20мс и 10мс соответственно. Кнопка ПУСК запускает генератор. Кнопка СТОП – выключает.
5. Разработать устройство управления:

– при размыкании одного из 4 двоичных ключей включается звуковой сигнал и мигание светодиода (длительность 1.5 с);
– при замыкании всех ключей входной двоичный код вводится с одного из портов ввода, сохраняется в памяти и выводиться на семисегментный индикатор.Начинает работать по кнопке ON, останов – по кнопке OFF.

6. Разработать генератор прямоугольных импульсов (меандра) с частотой fвых = 200 Гц, запустить генератор, нажав кнопку ГЕН; остановить генератор кнопкой ОСТ. Вывести на цифровой индикатор fвых.
7. Разработать генератор прямоугольных импульсов (меандра) с частотой fвых = 0,7 кГц. Запустить генератор, нажав кнопку ГЕН; остановить генератор кнопкой ОСТ. Вывести на цифровой индикатор fвых. Для генерирования выходной последовательности импульсов использовать программный метод организации временной задержки.
8. Разработать устройство измерения периода входного сигнала (скважность 2). Измерение начинается при замыкании двоичного ключа.

Максимальное время измерения 1с. Отобразить на цифровом индикаторе период сигнала в двоично- десятичном коде и выдать звуковой сигнал.

9. Разработать устройство измерения временного интервала между двумя событиями (включить конвейер, фотодатчик). Вывести на цифровой индикатор число деталей, прошедших через конвейер за время его включения. Фотодатчик формирует одиночный импульс в виде логического нуля при прохождении детали по конвейеру.

10. Разработать устройство для выполнения опроса сигнала на входе с интервалом 90 мс, после тридцатого обнаружения сигнала «логического нуля» отобразить на цифровом индикаторе информацию количества опроса сигнала и выдать звуковой сигнал и включить зеленый светодиод. Опрос начинается при нажатии кнопок ПУСК и НУЛЬ.

11. Разработать устройство для выполнения опроса сигнала на входе с интервалом 50 мс, после пятнадцатого обнаружения сигнала «логической единицы» отобразить на цифровом индикаторе информацию количества опроса сигнала и выдать прерывистый звуковой сигнал. Опрос начинается при нажатии кнопок ПУСК и ЕДИН.

12. Разработать устройство управления информацией. В память программ, начиная с адреса $9000, размещены коды символов текстовой строки: DEAR FRIEND! WELCOME TO KAZAN! Переписать символы в массив памяти данных. При нажатии кнопок «ВКЛ» и «ВЫВОД» вывести на цифровой индикатор коды цифр: 2018. Выдать звуковой сигнал, мигание светодиода.

13. Разработать логическое устройство управления. На линии порта в/в (5 разрядов) поступают цифровые сигналы I1,I2,I3,I4,I5 от двоичных датчиков. На цифровой индикатор вывести значения двоичных датчиков и значение функции I1&I3+I4&I2+I1&I5. Выдать звуковой сигнал.

14. Разработать логическое устройство управления. На линии порта ввода (6 разрядов) поступают цифровые сигналы I1,I2,I3,I4,I5 от двоичных датчиков (двоичные ключи). На цифровой индикатор вывести значения двоичных датчиков и значение функции I1&I2+I3&I4+I5&I6. Выдать звуковой сигнал.

15. Разработать устройство измерения длительности одиночного импульса, поступающего на вход микроконтроллера. Информацию о длительности импульса (до 9999мкс) вывести на цифровой индикатор. Запуск измерителя производиться нажатием кнопок «ПУСК» и «ИЗМЕРЕНИЕ».

16. Разработать микроконтроллерное устройство для подсчета числа импульсов, поступающих на вход микроконтроллера. Информацию о количестве импульсов (до 255) вывести на цифровой индикатор. Запуск производиться нажатием кнопок «ПУСК» и «СЧЕТ».

17. Разработать устройство для подсчета числа импульсов, поступающих на вход микроконтроллера за заданный промежуток времени t=300 мкс. Информацию о количестве импульсов (до 255) вывести на цифровой индикатор. Запуск производиться нажатием кнопок «ПУСК» и «СЧЕТ».

18. Разработать устройство управления термостатом по следующему алгоритму:

– определить текущее значение tтек. С датчика температуры;

– вычислить разность между фактическим и стандартным значением

tстанд (tфакт – tстанд) = Δt;

– если значение разности Δt > t1, то включить световой и звуковой сигналы;

– если Δt <t1 и t1 Δt > t2 , то включить только световой сигнал; – если Δt < 0, включить нагреватель; – если Δt > 0, включить вентилятор.
– вывести на индикацию текущее значение температуры.

19. Разработать устройство ввода. Ввести с 12-кнопочной клавиатуры десятичные цифры и сохранить их в ОЗУ микроконтроллера. После сохранения каждого числа выдавать звуковой сигнал с помощью пъезодинамика с заданной частотой f = 1 кГц. Найти среднее арифметическое чисел и вывести на цифровой индикатор. Ввод начать после нажатия кнопки «ВВОД», закончить после нажатия кнопки «СТОП».

20. Разработать устройство – световой автомат для управления составной светодиодной линейкой из восьми светодиодов. Устройство должно обеспечивать поочередное движение огня в двух направлениях. Переключение должно осуществляться при помощи двоичных ключей ВЛЕВО, ВПРАВО с выводом на цифровой индикатор кодов цифр 000 и FFF соответственно. Запуск выполняется кнопкой ПУСК.

Приложение 2

Структура содержания курсового проекта:

1. **Функциональная схема (прил.3)**

Функциональная схема, представленная в данномразделе,визуализирует архитектуру разрабатываемой системы. Она отображаетосновные блоки, включая клавиатуру, микроконтроллер ATmega328(блокобработки) и выходной индикатор. Данная схема обеспечивает пониманиепринципов функционирования системы, логики взаимодействия компонентови является основой для дальнейшей детализации при проектировании.

2. **Принципиалльная схема**

Принципиальная схема – это детальное графическое изображениевсехэлектрических связей и компонентов, составляющих устройство. Онаслужитосновой для понимания принципа работы и функциональных взаимосвязеймежду элементами

Она демонстрирует полную схему подключения компонентовустройства к ATmega328. На ней детально показаны электрическиесоединения, включая матричную клавиатуру 4×3, шестиразрядныйсемисегментный индикатор и пьезодинамик.

3. **Алгоритм работы**

Описывается блок-схема: настройка портов и пинов, блок схемы работы клавиш и т.д.

4. **Программа**

Приложение 3

Рис.1. Функциональная схема

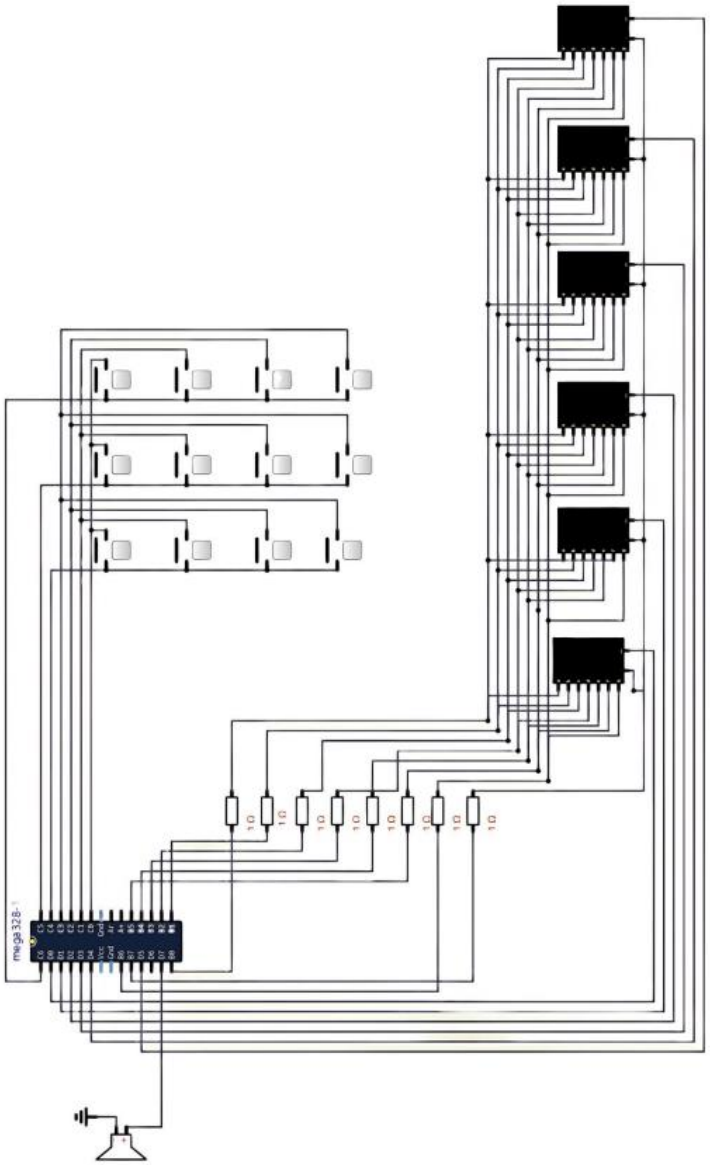


Рис.2. Принципиадльная схема

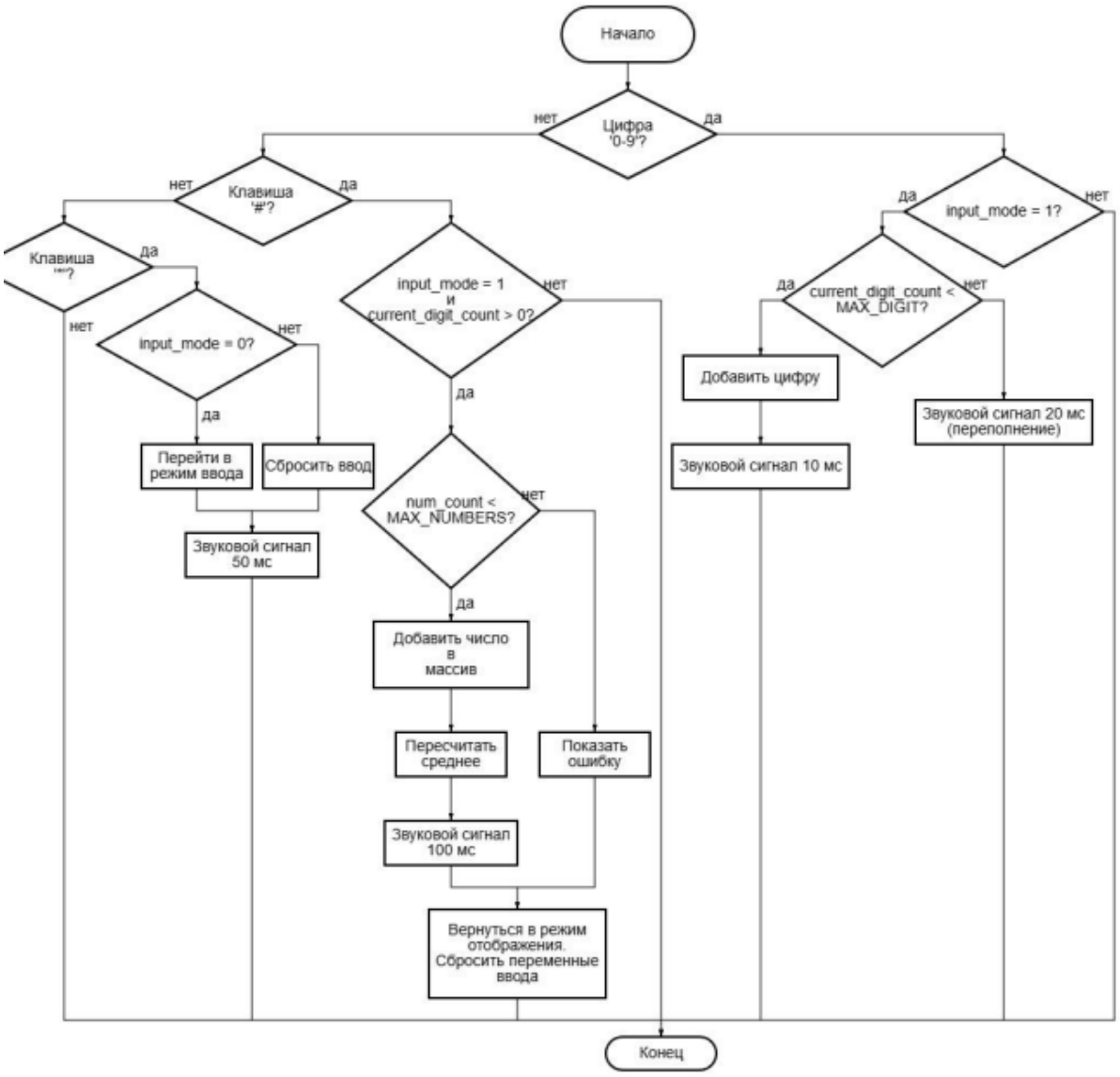


Рис.3. Блок схема работы клавиш

```
// ==================== НАСТРОЙКА ПОРТОВ ====================
void setup_ports(void) {
    // Настройка клавиатуры
    KEY_ROW_DDR |= (1 << KEY_ROW1) | (1 << KEY_ROW2) |
                   (1 << KEY_ROW3) | (1 << KEY_ROW4);
    KEY_ROW_PORT |= (1 << KEY_ROW1) | (1 << KEY_ROW2) |
                    (1 << KEY_ROW3) | (1 << KEY_ROW4);

    KEY_COL_DDR &= ~((1 << KEY_COL1) | (1 << KEY_COL2) | (1 << KEY_COL3));
    KEY_COL_PORT |= (1 << KEY_COL1) | (1 << KEY_COL2) | (1 << KEY_COL3);

    // Настройка 7-сегментного индикатора
    SEGMENT_DDR = 0xFF;          // Все пины порта B как выходы
    SEGMENT_PORT = 0x00;         // Выключить все сегменты

    DIGIT_DDR |= (1 << DIGIT_1) | (1 << DIGIT_2) | (1 << DIGIT_3) |
                 (1 << DIGIT_4) | (1 << DIGIT_5) | (1 << DIGIT_6);
    DIGIT_PORT |= (1 << DIGIT_1) | (1 << DIGIT_2) | (1 << DIGIT_3) |
                  (1 << DIGIT_4) | (1 << DIGIT_5) | (1 << DIGIT_6);

    // Настройка пина для пьезодинамика
    BUZZER_DDR |= (1 << BUZZER_PIN);
    BUZZER_PORT &= ~(1 << BUZZER_PIN);
}

// ==================== ЗВУКОВОЙ СИГНАЛ ====================
void beep(uint16_t duration_ms) {
    uint16_t i;
    uint16_t cycles = duration_ms * 8; // Приблизительно 8 циклов на 1 мс

    for (i = 0; i < cycles; i++) {
        BUZZER_PORT ^= (1 << BUZZER_PIN);
        _delay_us(62); // Приблизительно 500 Гц
    }
    BUZZER_PORT &= ~(1 << BUZZER_PIN);
}
```

Рис.4. Часть составленного кода для задания варианта 19

Учебное издание

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ЭЛЕКТРОТЕХНИКИ

Учебно-методическое пособие

Составитель
Садыков Марат Фердинантович

Кафедра теоретических основ электротехники КГЭУ

Редакторы *С. Н. Чемоданова, И. В. Краснова*
Технический редактор *И. В. Краснова*
Компьютерная верстка *И. В. Красновой*

Подписано в печать .
Формат 60×84/16. Усл. печ. л…... Уч.-изд. л. …..
Тираж экз. Заказ № /эл.

Редакционно-издательский отдел КГЭУ.
420066, г. Казань, ул. Красносельская, 51