МИРОВАЯ ЭЛЕКТРОНИКА
СЕРИЯ

А. В. Евстифеев

Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL

5-е издание

ДОДЭКА

Москва
Издательский дом «Додэка-XXI»
2008

УДК 621.316.544.1 (035.5)
ББК 32.844.1-04я2
Е26

Е26 **Евстифеев А.В.**
Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL, 5-е изд., стер. — М.: Издательский дом «Додэка-XXI», 2008. — 560 с.

ISBN 978-5-94120-220-1

Книга посвящена вопросам практического применения однокристальных микроконтроллеров AVR семейств Tiny и Mega фирмы ATMEL.

Рассмотрена архитектура, ее особенности. Приведены основные электрические параметры и временные характеристики. Подробно описано внутреннее устройство микроконтроллеров, системы команд, периферия, а также способы программирования с примерами реализации некоторых алгоритмов для конкретных цифровых устройств.

Книга предназначена для разработчиков радиоэлектронной аппаратуры, инженеров, студентов вузов и радиолюбителей.

УДК 621.316.544.1 (035.5)
ББК 32.844.1-04я2

ISBN 978-5-94120-220-1

Оглавление

Предисловие

Книга, которую вы держите в руках, — это вторая книга, посвященная микроконтроллерам AVR фирмы «Atmel». Эти 8-разрядные RISC-микроконтроллеры для встраиваемых приложений являются, пожалуй, наиболее интересным и прогрессивным направлением, развиваемым фирмой. Микроконтроллеры этой серии представляют собой мощный инструмент, прекрасную основу для создания современных высокопроизводительных и экономичных встраиваемых контроллеров многоцелевого назначения.

Популярность микроконтроллеров AVR постоянно увеличивается. Не последнюю роль в этом играет соотношение показателей «цена/быстродействие/энергопотребление», являющееся одним из лучших на рынке 8-разрядных микроконтроллеров. Кроме того, постоянно растет число выпускаемых сторонними производителями разнообразных программных и аппаратных средств поддержки разработок устройств на их основе. Все это позволяет говорить о микроконтроллерах AVR как о новом индустриальном стандарте среди 8-разрядных микроконтроллеров общего применения.

В рамках единой базовой архитектуры микроконтроллеры AVR подразделяются на три семейства:

- Classic AVR;
- Mega AVR;
- Tiny AVR.

Микроконтроллеры семейства Classic были описаны в первой книге серии, а данная книга посвящена двум последним семействам — Tiny и Mega.

Микроконтроллеры семейства Tiny имеют небольшие объемы памяти программ (1...2 Кбайта) и весьма ограниченную периферию. Практически все они выпускаются в 8-выводных корпусах и предна-

значены для т. н. «бюджетных» решений, принимаемых в условиях жестких финансовых ограничений. Область применения этих микроконтроллеров — интеллектуальные датчики различного назначения (контрольные, пожарные, охранные), игрушки, зарядные устройства, различная бытовая техника и другие подобные устройства.

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

Микроконтроллеры обоих семейств поддерживают несколько режимов пониженного энергопотребления, имеют блок прерываний, сторожевой таймер и допускают программирование непосредственно в готовом устройстве.

В предлагаемой вашему вниманию книге представлена вся информация, необходимая для изучения микроконтроллеров AVR семейств Tiny и Mega. Однако следует заметить, что всеобъемлющим справочником данная книга не является, хотя и написана на основе документации, предоставляемой фирмой «Atmel». Поэтому, прежде чем приступить к практическому использованию рассматриваемых микроконтроллеров, настоятельно рекомендуется обратиться к официальной информации, размещенной на Web-сайтах фирмы (www.atmel.com, www.atmel.ru).

Часть 1
Микроконтроллеры семейства Tiny

Глава 1. **Знакомство с семейством Tiny**

1.1. Общие сведения

Как и все микроконтроллеры AVR фирмы «Atmel», микроконтроллеры семейства Tiny являются 8-разрядными микроконтроллерами, предназначенными для встраиваемых приложений. Они изготавливаются по малопотребляющей КМОП-технологии, которая в сочетании с усовершенствованной RISC-архитектурой позволяет достичь наилучшего соотношения быстродействие/энергопотребление. Удельное быстродействие этих микроконтроллеров может достигать значения 1 MIPS/МГц (1 миллион операций в секунду на 1 МГц тактовой частоты). Модели микроконтроллеров семейства Tiny и их основные параметры приведены в приложениях 1 и 4. Микроконтроллеры описываемого семейства предназначены в первую очередь для низкостоимостных («бюджетных») приложений и соответственно являются самыми дешевыми из всех микроконтроллеров AVR. Важной особенностью этих микроконтроллеров является эффективное использование выводов кристалла, например, в 8-выводном корпусе все выводы (кроме, разумеется, выводов питания) могут использоваться в качестве линий ввода/вывода.

1.2. Отличительные особенности

Перечислим вкратце основные особенности микроконтроллеров семейства Tiny:

- возможность вычислений со скоростью до 1 MIPS/МГц;
- FLASH-память программ объемом 1...2 Кбайт (число циклов стирания/записи не менее 1000);
- оперативная память (статическое ОЗУ) объемом 1...2 Кбайт;

- память данных на основе ЭСППЗУ (EEPROM) объемом до 64 байт (число циклов стирания/записи не менее 100000);
- возможность защиты от внешнего чтения и модификации памяти программ и данных (в EEPROM);
- возможность программирования непосредственно в системе через последовательный интерфейс*;
- различные способы синхронизации: встроенный генератор с внутренней или внешней времязадающей *RC*-цепочкой; встроенный генератор с внешним резонатором (пьезокерамическим или кварцевым); внешний сигнал синхронизации;
- наличие двух или трех режимов пониженного энергопотребления;
- некоторые модели микроконтроллеров могут работать при пониженном до 1.8 В напряжении питания.

1.3. Характеристики процессора

Основными характеристиками центрального процессора микроконтроллеров рассматриваемого семейства являются:

- полностью статическая архитектура; минимальная тактовая частота равна нулю;
- АЛУ подключено непосредственно к регистрам общего назначения;
- большинство команд выполняются за один машинный цикл;
- многоуровневая система прерываний; поддержка очереди прерываний;
- 5...8 источников прерываний (из них до 2-х внешних)**;
- трехуровневый аппаратный стек.

1.4. Характеристики подсистемы ввода/вывода

Основными характеристиками подсистемы ввода/вывода являются:

- программное конфигурирование и выбор портов ввода/вывода;
- выводы могут быть запрограммированы как входные или как выходные независимо друг от друга;
- входные буферы с триггером Шмитта на всех выводах;
- возможность подключения к входам внутренних подтягивающих резисторов (сопротивление резисторов составляет 35...120 кОм).

* Не во всех моделях.

** Зависит от конкретной модели микроконтроллера.

1.5. Периферийные устройства

Набор периферийных устройств, имеющихся в составе того или иного микроконтроллера, зависит от конкретной модели и может быть определен по сводной таблице, приведенной в Приложении 1. Вообще же в составе микроконтроллеров семейства встречаются следующие периферийные устройства:

- 8-разрядный таймер/счетчик с предделителем (таймер T0)*;
- второй 8-разрядный таймер/счетчик с предделителем (таймер T1)**;
- сторожевой таймер WDT*;
- одноканальный генератор сигнала с ШИМ разрядностью 8 бит (один из режимов работы таймера T1);
- аналоговый компаратор*;
- 10-разрядный АЦП (4 канала);
- аппаратный модулятор.

1.6. Архитектура ядра

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

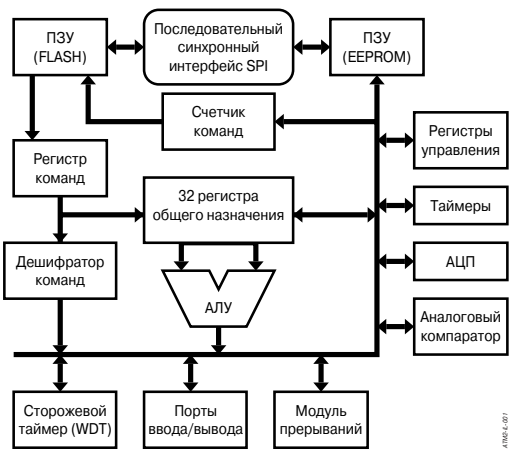


Рис. 1.1. Архитектура ядра микроконтроллеров AVR семейства Tiny

* Присутствует во всех моделях.

** Присутствует только в модели ATtiny 15L.

Арифметико-логическое устройство (АЛУ), выполняющее все вычисления, подключено непосредственно к 32-м рабочим регистрам, объединенным в регистровый файл. Благодаря этому АЛУ выполняет одну операцию (чтение содержимого регистров, выполнение операции и запись результата обратно в регистровый файл) за один машинный цикл. Кроме того, в микроконтроллерах семейства Tiny каждая из команд занимает только одну ячейку памяти программ.

В микроконтроллерах AVR реализована Гарвардская архитектура, которая характеризуется раздельной памятью программ и данных, каждая из которых имеет собственные шины доступа к ним. Такая организация позволяет одновременно работать как с памятью программ, так и с памятью данных. Разделение шин доступа позволяет использовать для каждого типа памяти шины различной разрядности, а также реализовать конвейеризацию. Конвейеризация заключается в том, что во время исполнения текущей команды производится выборка из памяти и дешифрация кода следующей команды.

В отличие от RISC-микроконтроллеров других фирм, в микроконтроллерах AVR используется 2-уровневый конвейер, а длительность машинного цикла составляет всего один период кварцевого резонатора. В результате, при более низкой тактовой частоте они могут обеспечивать ту же производительность, что и RISC-микроконтроллеры других фирм.

1.7. Цоколевка и описание выводов

В семейство Tiny входит в общей сложности 8 моделей микроконтроллеров, которые составляют 4 группы:

- ATtiny11, ATtiny11L **(Рис. 1.2)** имеют FLASH-память программ объемом 1 Кбайт. Максимальное количество контактов ввода/вывода равно 6 (вывод 1 может использоваться только как входной);
- ATtiny12, ATtiny12L, ATtiny12V **(Рис. 1.3)** имеют FLASH-память программ объемом 1 Кбайт и EEPROM-память данных объемом 64 байт. Максимальное количество контактов ввода/вывода равно 6;
- ATtiny15L **(Рис. 1.4)** имеет FLASH-память программ объемом 1 Кбайт и EEPROM-память данных объемом 64 байт. Максимальное количество контактов ввода/вывода равно 6;
- ATtiny28L, ATtiny28V **(Рис. 1.5)** имеют FLASH-память программ 2 Кбайт. Количество контактов ввода/вывода равно 19 (из них 11 — контакты ввода/вывода общего назначения, а 8 — входные контакты).

Пока книга готовилась к печати, фирма «Atmel» выпустила еще две модели микроконтроллеров семейства — ATtiny26/ATtiny26L. Эти микроконтроллеры имеют FLASH-память программ объемом 2 Кбайт, ОЗУ

объемом 128 байт и EEPROM-память данных объемом 128 байт. Количество контактов ввода/вывода в этих моделях равно 16. В данной книге эти модели по понятным причинам не описываются, однако основные их параметры приведены в Приложении 1.

Для сравнения в **Табл. 1.1** приводятся основные параметры микроконтроллеров, такие, как объем памяти (программ и данных), количество контактов ввода/вывода, тип корпуса, диапазон рабочих частот и напряжение питания. Полная информация по каждой модели приведена в Приложении 1. Дополнительно следует отметить, что все микроконтроллеры семейства Tiny выпускаются как в коммерческом (диапазон рабочих температур 0...+70°C), так и в промышленном (диапазон рабочих температур –40...+85°C) исполнениях.

Таблица 1.1. **Основные параметры микроконтроллеров AVR семейства Tiny**

Обозначение	Память программ (FLASH) [Кбайт]	Память данных (EEPROM) [байт]	Количество линий ввода/вывода	Напряжение питания [В]	Тактовая частота [МГц]	Тип корпуса
ATtiny11	1	—	6	4.0...5.5	0...6	DIP-8
						SOIC-8
ATtiny11L	1	—	6	2.7... 5.5	0...2	DIP-8
						SOIC-8
ATtiny12	1	64	6	4.0...5.5	0...8	DIP-8
						SOIC-8
ATtiny12L	1	64	6	2.7... 5.5	0...4	DIP-8
						SOIC-8
ATtiny12V	1	64	6	1.8...5.5	0...1.2	DIP-8
						SOIC-8
ATtiny15L	1	64	6	2.7...5.5	0...1.2	DIP-8
						SOIC-8
ATtiny28L	2	—	19	2.7...5.5	0...4	DIP-28
						TQFP-32
						MLF-32
ATtiny28V	2	—	19	1.8...5.5	0...1.2	DIP-28
						TQFP-32
						MLF-32

В **Табл. 1.2...1.5** для каждой группы микроконтроллеров приведены названия выводов и указаны их функции (как основные, так и дополнительные). 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 – выводы питания.

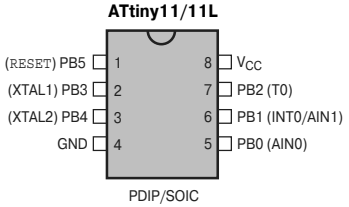


Рис. 1.2. Расположение выводов (вид сверху) моделей ATtiny11/11L

Таблица 1.2. **Описание выводов модели ATtiny11/11L**

Обозначение	Номер вывода	Тип вывода	Описание
PB0 (AIN0)	5	I/O	0-й разряд порта B (Положительный вход компаратора)
PB1 (INT0/AIN1)	6	I/O	1-й разряд порта B (Вход внешнего прерывания/ Отрицательный вход компаратора)
PB2 (T0)	7	I/O	2-й разряд порта B (Вход внешнего тактового сигнала таймера/счетчика T0)
PB3 (XTAL1)	2	I/O	3-й разряд порта B (Вход тактового генератора)
PB4 (XTAL2)	3	I/O	4-й разряд порта B (Выход тактового генератора)
PB5 ($\overline{\text{RESET}}$)	1	I	5-й разряд порта B (Вход сброса)
GND	4	P	Общий вывод
V_{CC}	8	P	Вывод источника питания

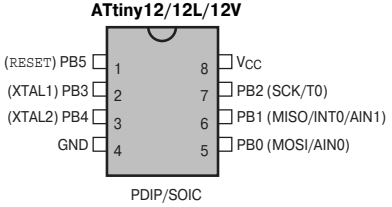


Рис. 1.3. Расположение выводов (вид сверху) моделей ATtiny12/12L/12V

Таблица 1.3. **Описание выводов модели ATtiny12/12L/12V**

Обозначение	Номер вывода	Тип вывода	Описание
PB0 (MOSI/AIN0)	5	I/O	0-й разряд порта B (Вход данных при программировании/Положительный вход компаратора)
PB1 (MISO/INT0/ AIN1)	6	I/O	1-й разряд порта B (Выход данных при программировании/Вход внешнего прерывания/ Отрицательный вход компаратора)
PB2 (SCK/T0)	7	I/O	2-й разряд порта B (Вход тактового сигнала при программировании/Вход внешнего тактового сигнала таймера/счетчика T0)
PB3 (XTAL1)	2	I/O	3-й разряд порта B (Вход тактового генератора)
PB4 (XTAL2)	3	I/O	4-й разряд порта B (Выход тактового генератора)
PB5 ($\overline{\text{RESET}}$)	1	I/O	5-й разряд порта B, тип выхода — открытый коллектор (Вход сброса)
GND	4	P	Общий вывод
V_{CC}	8	P	Вывод источника питания

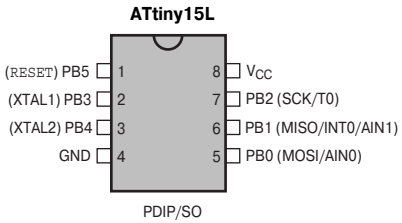


Рис. 1.4. Расположение выводов (вид сверху) модели ATtiny15L

Таблица 1.4. **Описание выводов модели ATtiny15L**

Обозначение	Номер вывода	Тип вывода	Описание
PB0 (AIN0/ AREF/ MOSI)	5	I/O	0-й разряд порта B (Положительный вход компаратора/Вход опорного напряжения для АЦП/Вход данных при программировании)
PB1 (AIN1/ OC1A/MISO)	6	I/O	1-й разряд порта B (Отрицательный вход компаратора/Выход таймера/счетчика T1 (режимы Compare, PWM)/Выход данных при программировании)
PB2 (ADC1/ T0/INT0/ SCK)	7	I/O	2-й разряд порта B (Вход АЦП/Вход внешнего тактового сигнала таймера/счетчика T0/Вход внешнего прерывания/Вход тактового сигнала при программировании)
PB3 (ADC2)	3	I/O	3-й разряд порта B (Вход АЦП)
PB4 (ADC3)	2	I/O	4-й разряд порта B (Вход АЦП)
PB5 (ADC0/ $\overline{\text{RESET}}$)	1	I/O	5-й разряд порта B (Вход АЦП/Вход сброса)
GND	4	P	Общий вывод
V_{CC}	8	P	Вывод источника питания

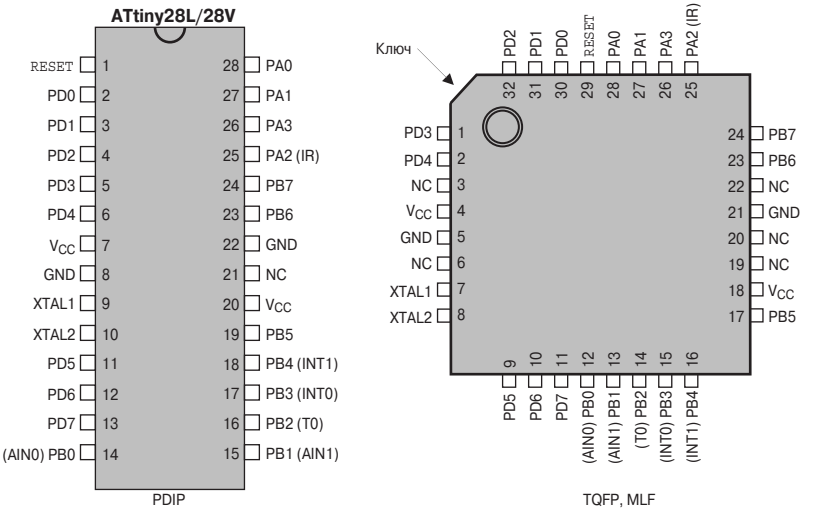


Рис. 1.5. Расположение выводов (вид сверху) моделей ATtiny28L/28V

Таблица 1.5. **Описание выводов моделей ATtiny28L/28V**

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP, MLF		
XTAL1	9	7	I	Вход тактового генератора
XTAL2	10	8	O	Выход тактового генератора
$\overline{\text{RESET}}$	1	29	I	Вход сброса
PA0	28	28	I/O	0-й разряд порта A
PA1	27	27	I/O	1-й разряд порта A
PA2 (IR)	25	25	O	2-й разряд порта A (Выходной контакт сповышенной нагрузочной способностью)
PA3	26	26	I/O	3-й разряд порта A
PB0 (AIN0)	14	12	I/O	0-й разряд порта B (Положительный вход компаратора)
PB1 (AIN1)	15	13	I/O	1-й разряд порта B (Отрицательный вход компаратора)
PB2 (T0)	16	14	I/O	2-й разряд порта B (Вход внешнего тактового сигнала таймера/счетчика T0)
PB3 (INT0)	17	15	I/O	3-й разряд порта B (Вход внешнего прерывания)
PB4 (INT1)	18	16	I/O	4-й разряд порта B (Вход внешнего прерывания)
PB5	19	17	I/O	5-й разряд порта B
PB6	23	23	I/O	6-й разряд порта B
PB7	24	24	I/O	7-й разряд порта B
PD0	2	30	I/O	8-разрядный двунаправленный порт ввода/вывода D
PD1	3	31	I/O	
PD2	4	32	I/O	
PD3	5	1	I/O	
PD4	6	2	I/O	
PD5	11	9	I/O	
PD6	12	10	I/O	
PD7	13	11	I/O	
GND	8, 22	5, 21	P	Общий вывод
V_{CC}	7, 20	4	P	Вывод источника питания
NC	21	3, 6, 19, 20, 22	—	Не используются

Глава 2. Архитектура микроконтроллеров семейства Tiny

2.1. Общие сведения

Микроконтроллеры AVR семейства Tiny являются 8-разрядными микроконтроллерами с RISC-архитектурой. Они имеют электрически стираемую FLASH-память программ (ряд моделей имеет также энергонезависимую EEPROM-память данных), а также разнообразные периферийные устройства. Состав этих устройств меняется от модели к модели, более того, одно и то же устройство в разных моделях использует различные ресурсы микроконтроллера (в частности, различные выводы). В то же время некоторые периферийные устройства присутствуют во всех микроконтроллерах семейства: сторожевой таймер, аналоговый компаратор, 8-разрядный таймер/счетчик реального времени и, естественно, порты ввода/вывода.

На **Рис. 1.6** и **Рис. 1.7** приведены структурные схемы микроконтроллеров ATtiny11/11L и ATtiny12/12L/12V соответственно. Их отличительные особенности:

- 6-разрядный порт ввода/вывода;
- 3-уровневый аппаратный стек;
- встроенный тактовый *RC*-генератор;
- возможность подключения внешнего резонатора;
- использование тактового генератора сторожевого таймера в качестве системного (только ATtiny11x).

Структурная схема микроконтроллера ATtiny15L приведена на **Рис. 1.8**. Его отличительные особенности:

- EEPROM-память данных объемом 64 байт;
- 6-разрядный порт ввода/вывода;
- возможность работы только от встроенного тактового *RC*-генератора;
- два 8-разрядных таймера/счетчика;
- 4-канальный АЦП.

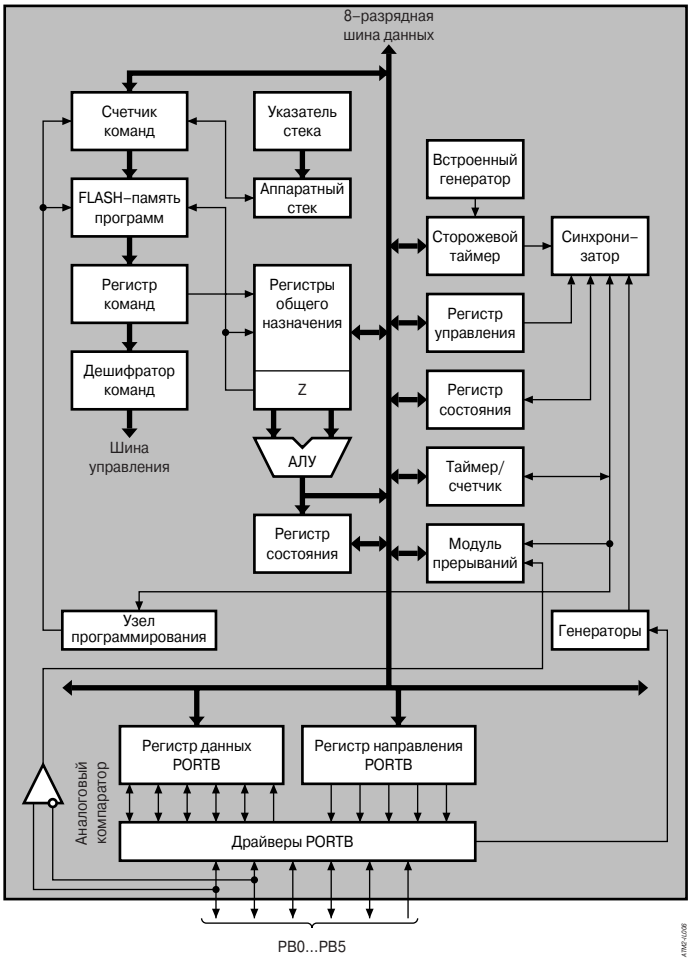


Рис. 1.6. Структурная схема микроконтроллеров ATtiny11/11L

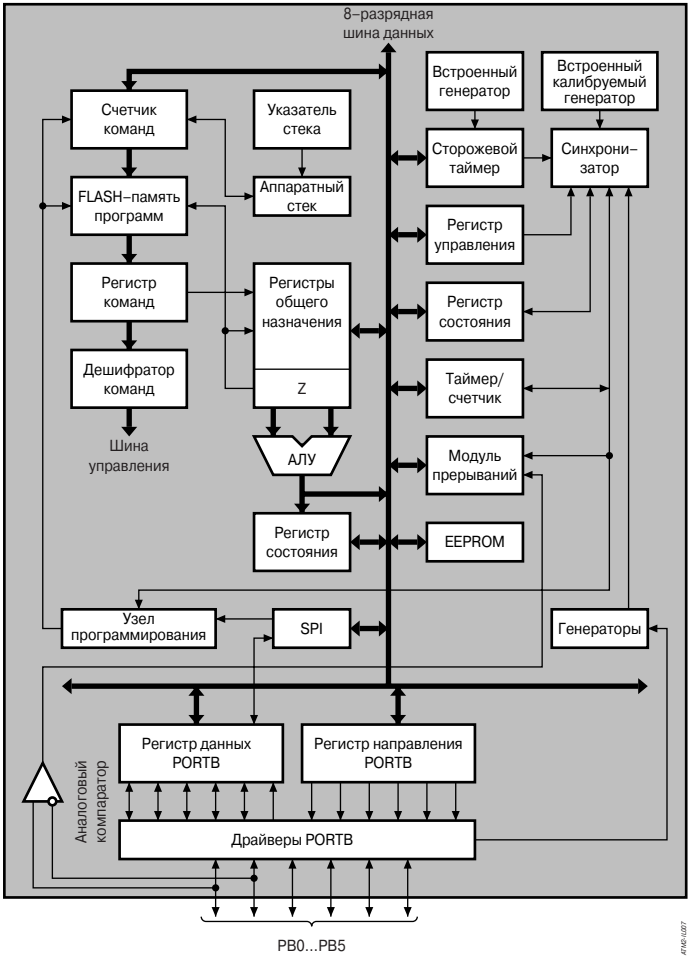


Рис. 1.7. Структурная схема микроконтроллеров ATtiny12/12L/12V

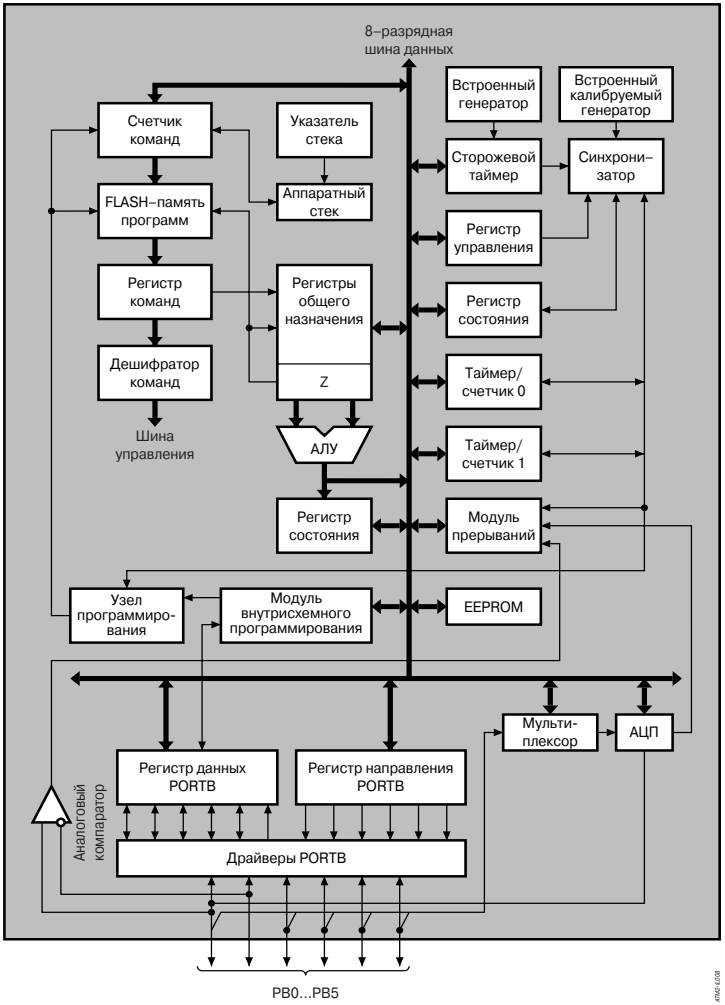


Рис. 1.8. Структурная схема микроконтроллера ATtiny15L

Структурная схема микроконтроллеров ATtiny28L/28V приведена на **Рис. 1.9**. Их отличительные особенности:

- 3 порта ввода/вывода: порт A (4-разрядный), порт B (8-разрядный) и порт D (8-разрядный);
- встроенный тактовый *RC*-генератор;

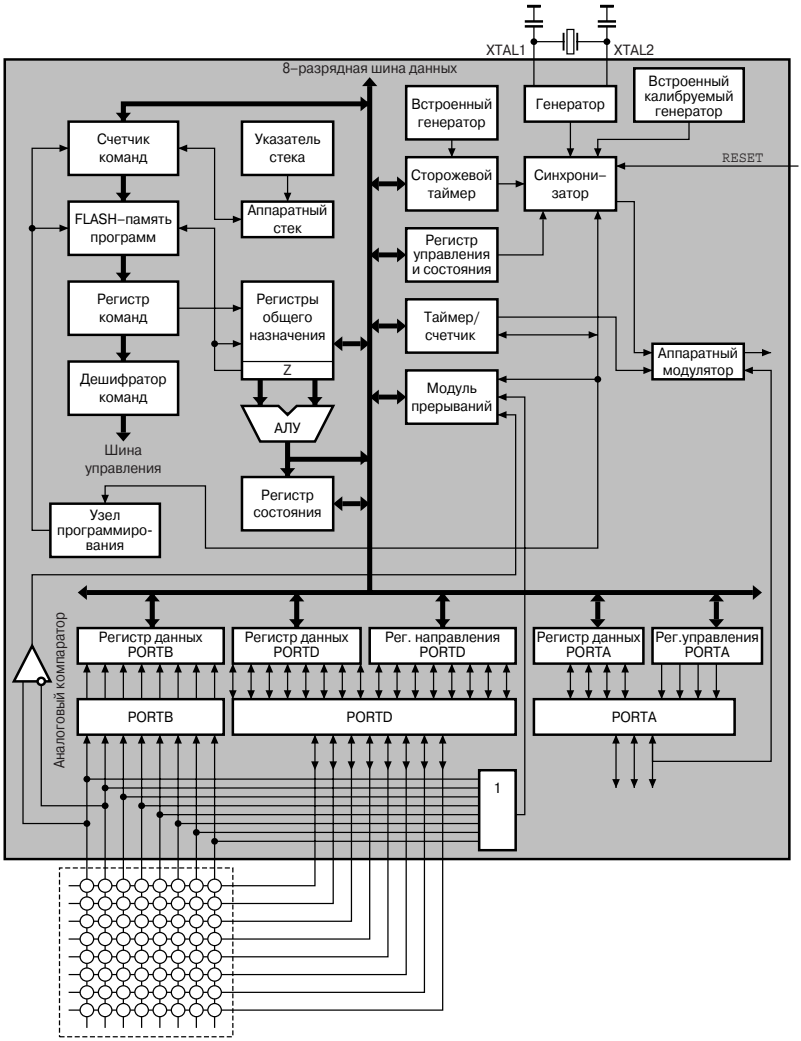


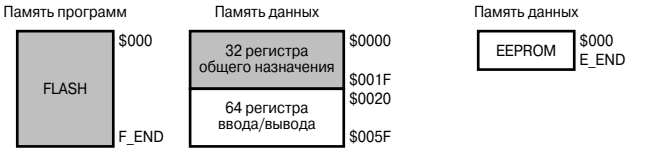
Рис. 1.9. Структурная схема микроконтроллеров ATtiny28L/28V

- возможность подключения внешнего резонатора;
- два входа внешних прерываний;
- наличие аппаратного модулятора для управления светодиодным индикатором.

2.2. Организация памяти

Организация памяти микроконтроллеров AVR семейства Tiny выполнена по схеме Гарвардского типа, в которой разделены не только адресные пространства памяти программ и памяти данных, но также и шины доступа к ним. Память данных состоит из двух областей: регистровая память и память на основе EEPROM. Каждая область расположена в своем адресном пространстве.

Обобщенная карта памяти микроконтроллеров AVR семейства Tiny приведена на **Рис. 1.10**.



Модель	Память программ (FLASH)		Память данных (EEPROM)	
	Верхняя граница (F_END)	Объем, слов	Верхняя граница (F_END)	Объем, байт
ATiny11/11L	$3FF	1 K	-	-
ATiny12/12L/12V	$3FF	1 K	$03F	64
ATiny15L	$3FF	1 K	$03F	64
ATiny28L/28V	$7FF	1 K	-	-

Рис. 1.10. Карта памяти микроконтроллеров семейства Tiny

Обратите внимание на следующие моменты. Поскольку микроконтроллеры AVR имеют 16-разрядную систему команд, объем памяти программ на рисунке указан не в байтах, а в 16-битных словах. Символ «$» перед числом означает, что это число записано в шестнадцатиричной системе счисления.

2.2.1. Память программ

Как следует из названия, память программ предназначена для хранения команд, управляющих функционированием микроконтроллера. В памяти программ хранятся также различные константы, не меняющиеся во время работы программы. Как уже было сказано, память программ представляет собой электрически стираемое ППЗУ (FLASH-ПЗУ). Поскольку все команды занимают в памяти по 16 бит, память программ имеет 16-разрядную организацию. Соответственно объем памяти микроконтроллеров семейства составляет 512...1024 16-битных слова.

Для адресации памяти программ используется счетчик команд (PC — Program Counter). Размер счетчика команд составляет 9 или 10 разрядов в зависимости от объема адресуемой памяти.

По адресу $000 памяти программ находится вектор сброса. После инициализации (сброса) микроконтроллера выполнение программы начинается с этого адреса (по этому адресу рекомендуется размещать команду относительного перехода к инициализационной части программы). Начиная с адреса $001 располагается таблица векторов прерываний. Размер этой области зависит от модели микроконтроллера и составляет от 4 (адреса $001...$004) до 8 (адреса $001...$008) векторов (подробнее о распределении области векторов прерывания см. раздел 3.5).

При возникновении прерывания после сохранения в стеке текущего значения счетчика команд происходит выполнение команды, расположенной по адресу соответствующего вектора. Поэтому по этим адресам располагаются команды относительного перехода к подпрограммам обработки прерываний. Ниже приведен типичный листинг начала программы для модели ATtiny28:

```
Address   Labels   Code                 Comments
$000               rjmp RESET           ;Обработчик сброса
$001               rjmp EXT_INT0        ;Обработчик внешнего
                                        ;прерывания
$002               rjmp EXT_INT1        ;Обработчик внешнего
                                        ;прерывания
$003               rjmp LOW_LEVEL       ;Обработчик прерывания
                                        ;от порта B
$004               rjmp TIM_OVF0        ;Обработчик прерывания
                                        ;от таймера T0
$005               rjmp ANA_COMP        ;Обработчик прерывания
                                        ;от аналогового компаратора
$006      MAIN:    ldi r16,low(RAMEND)  ;Начало основной программы
                   out SPL,r16
                   <инструкция> xxx
```

Если в программе прерывания не используются (запрещены), то основная программа может начинаться непосредственно с адреса $001.

Следует отметить, что 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 имеется специальная команда — LPM Rd, Z. Адрес, по которому производится чтение, перед использованием этой команды должен быть загружен в индексный регистр Z (см. далее). При этом старшие 15 разря-

дов содержимого регистра будут определять адрес слова, а младший разряд будет определять, какой из байтов будет прочитан: «0» — младший байт, «1» — старший байт **(Рис. 1.11)**.

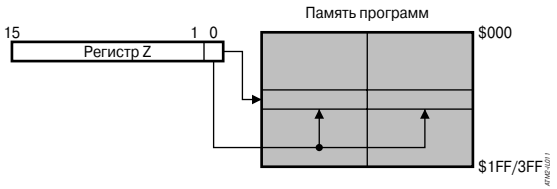


Рис. 1.11. Косвенная адресация памяти программ

В заключение следует отметить, что FLASH-ПЗУ, используемое в микроконтроллерах AVR, рассчитано как минимум на 1000 циклов стирания/записи.

2.2.2. Память данных

Память данных микроконтроллеров семейства Tiny разделена на две части: регистровая память и энергонезависимое ЭСППЗУ (EEPROM). Внутреннее статическое ОЗУ в микроконтроллерах семейства Tiny отсутствует.

Регистровая память включает 32 регистра общего назначения (РОН), объединенных в т. н. «файл» и служебные регистры ввода/вывода (РВВ). Размер регистровой памяти фиксирован и для всех моделей составляет 96 байт, соответственно под РОН отводится 32 байт, а под РВВ — 64 байт.

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

Для долговременного хранения различной информации, которая может изменяться в процессе функционирования готовой системы (калибровочные константы, серийные номера, ключи и т. п.) в моделях ATtiny12x и ATtiny15L может быть использована EEPROM-память. Ее объем составляет 64 байт. Эта память расположена в отдельном адресном пространстве, а доступ к ней осуществляется с помощью определенных регистров ввода/вывода.

2.2.2.1. Регистры общего назначения

Все регистры общего назначения объединены в файл, структура которого показана на **Рис. 1.12**. Как уже было сказано, в микроконтроллерах AVR все 32 РОН непосредственно доступны АЛУ, в отличие от микроконтроллеров других фирм, в которых имеется только один такой регистр — рабочий регистр W (аккумулятор). Благодаря этому любой РОН может использоваться во всех командах и как операнд-источник, и как операнд-приемник. Исключение составляют лишь пять арифметических и логических команд, выполняющих действия между константой и регистром (SBCI, SUBI, CPI, ANDI, ORI), а также команда загрузки константы в регистр (LDI). Эти команды могут обращаться только ко второй половине регистров (R16...R31).

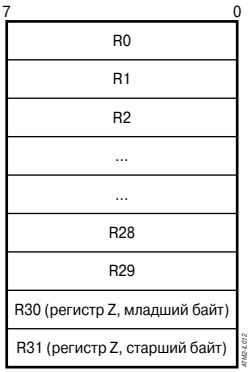


Рис. 1.12. Структура файла регистров общего назначения

Два старших регистра общего назначения формируют 16-разрядный индексный регистр Z, который используется в качестве указателя при косвенной адресации памяти программ и памяти данных. Так как объем адресуемой памяти данных составляет всего 32 байт, при обращении к ней используется только младший байт (регистр R30). Содержимое старшего байта индексного регистра (регистр R31) при косвенной адресации памяти данных автоматически очищается процессором. Более подробно использование этого регистра описано в подразделе 2.2.2.3.

2.2.2.2. Регистры ввода/вывода

Регистры ввода/вывода располагаются в так называемом пространстве ввода/вывода размером 64 байт. Все РВВ можно разделить на две группы: служебные регистры микроконтроллера и регистры, относящиеся к периферийным устройствам (в том числе порты ввода/вывода). Размер каждого регистра — 8 бит.

Распределение адресов пространства ввода/вывода зависит от конкретной модели микроконтроллера, т. к. разные модели имеют различный состав периферийных устройств и соответственно разное количество регистров. Размещение РВВ в адресном пространстве ввода/вывода для всех моделей семейства приведено в **Табл. 1.6**. Прочерк в таблице означает, что для данной модели этот адрес зарезервирован, но запись по этому адресу запрещена. Все регистры будут подробно описаны в соответствующих главах книги, а в этой главе будут рассмотрены только служебные регистры, общие для всех микроконтроллеров семейства.

Таблица 1.6. **РВВ микроконтроллеров семейства Tiny**

Название	Функция	Адрес	ATtiny			
			11x	12x	15L	28x
SREG	Регистр состояния	$3F	♦	♦	♦	♦
GIMSK	Общий регистр маски прерываний	$3B	♦	♦	♦	—
GIFR	Общий регистр флагов прерываний	$3A	♦	♦	♦	—
TIMSK	Регистр маски прерываний от таймера/счетчика	$39	♦	♦	♦	—
TIFR	Регистр флагов прерываний от таймера/счетчика	$38	♦	♦	♦	—
MCUCR	Общий регистр управления микроконтроллером	$35	♦	♦	♦	—
MCUSR	Регистр состояния микроконтроллера	$34	♦	♦	♦	—
TCCR0	Регистр управления таймером/счетчиком T0	$33	♦	♦	♦	—
TCNT0	Счетный регистр таймера/счетчика T0 (8-разрядный)	$32	♦	♦	♦	—
OSCCAL	Регистр калибровки тактового генератора	$31	—	♦	♦	—
TCCR1	Регистр управления таймером/счетчиком T1	$30	—	—	♦	—
TCNT1	Счетный регистр таймера/счетчика T1	$2F	—	—	♦	—
OCR1A	Регистр совпадения A таймера/счетчика T1	$2E	—	—	♦	—
OCR1B	Регистр совпадения B таймера/счетчика T1	$2D	—	—	♦	—
SFIOR	Регистр специальных функций	$2C	—	—	♦	—
WDTCR	Регистр управления сторожевым таймером	$21	♦	♦	♦	—
EEAR	Регистр адреса EEPROM	$1E	—	♦	♦	—
EEDR	Регистр данных EEPROM	$1D	—	♦	♦	—
EECR	Регистр управления EEPROM	$1C	—	♦	♦	—
PORTA	Регистр данных порта A	$1B	—	—	—	♦
PACR	Регистр управления порта A	$1A	—	—	—	♦
PINA	Выводы порта A	$19	—	—	—	♦

Продолжение таблицы 1.6

Название	Функция	Адрес	ATtiny			
			11x	12x	15L	28x
PORTB	Регистр данных порта B	$18	♦	♦	♦	—
DDRB	Регистр направления данных порта B	$17	♦	♦	♦	—
PINB	Выводы порта B	$16	♦	♦	♦	♦
PORTD	Регистр данных порта D	$12	—	—	—	♦
DDRD	Регистр направления данных порта D	$11	—	—	—	♦
PIND	Выводы порта D	$10	—	—	—	♦
ACSR	Регистр управления и состояния аналогового компаратора	$08	♦	♦	♦	♦
ADMUX	Регистр управления мультиплексором АЦП	$07	♦	—	—	—
MCUCS	Регистр управления и состояния микроконтроллера		—	—	—	♦
ADCSR	Регистр управления и состояния АЦП	$06	♦	—	—	—
ICR	Регистр управления прерываниями		—	—	—	♦
ADCH	Регистр данных АЦП (старший байт)	$05	♦	—	—	—
IFR	Регистр флагов прерываний		—	—	—	♦
ADCL	Регистр данных АЦП (младший байт)	$04	♦	—	—	—
TCCR0	Регистр управления таймером/счетчиком T0		—	—	—	♦
TCNT0	Счетный регистр таймера/счетчика T0 (8-разрядный)	$03	—	—	—	♦
MODCR	Регистр управления модулятором	$02	—	—	—	♦
WDTCR	Регистр управления сторожевым таймером	$01	—	—	—	♦
OSCCAL	Регистр калибровки тактового генератора	$00	—	—	—	♦

К любому регистру ввода/вывода можно обратиться с помощью команд IN и OUT, выполняющих пересылку данных между одним из 32-х РОН и пространством ввода/вывода. Кроме того, имеются 4 команды поразрядного доступа, использующие в качестве операндов регистры ввода/вывода: команды установки/сброса отдельного бита (SBI и CBI) и команды проверки состояния отдельного бита (SBIS и SBIC). Однако указанные команды могут обращаться только к 1-й половине регистров ввода/вывода (адреса $00...$1F).

SREG (регистр состояния)

Регистр состояния SREG (Status REGistr) располагается по адресу $3F. Этот регистр является, по сути дела, набором флагов, показывающих текущее состояние микроконтроллера. Эти флаги автоматически устанавливаются в «1» или в «0» при наступлении определенных событий (в соответствии с результатом выполнения команд). Все разряды этого регистра доступны как для чтения, так и для записи в любой момент времени; после сброса микроконтроллера все разряды регистра сбрасываются в «0». Содержимое этого регистра показано ниже на **Рис. 1.13**, а его описание приведено в **Табл. 1.7**.

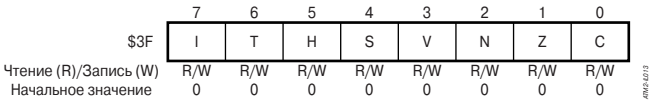


Рис. 1.13. Формат регистра состояния SREG

Таблица 1.7. **Разряды регистра состояния SREG**

Разряд	Название	Описание
7	I	**Общее разрешение прерываний.** Для разрешения прерываний этот флаг должен быть установлен в «1». Разрешение/запрещение отдельных прерываний производится установкой или сбросом соответствующих разрядов регистров масок прерываний (см. Главу 4). Если флаг сброшен (0), то прерывания запрещены независимо от состояния этих регистров. Флаг сбрасывается аппаратно после входа в прерывание и восстанавливается командой RETI для разрешения обработки следующих прерываний
6	T	**Хранение копируемого бита.** Этот разряд регистра используется в качестве источника или приемника командами копирования битов BLD (Bit LoaD) и BST (Bit STore). Заданный разряд любого РОН может быть скопирован в этот разряд командой BST или установлен в соответствии с содержимым данного разряда командой BLD
5	H	**Флаг половинного (Half) переноса.** Этот флаг устанавливается в «1», если имел место перенос из младшей половины байта (из 3-го разряда в 4-й), или заем из старшей половины байта при выполнении некоторых арифметических операций
4	S	**Флаг знака (Sign).** 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 (Overflow) дополнительного кода**. Этот флаг устанавливается в «1» при переполнении разрядной сетки знакового результата. Используется при работе со знаковыми числами (представленными в дополнительном коде). Более подробно — см. описание системы команд
2	N	**Флаг отрицательного (Negative) значения**. Этот флаг устанавливается в «1», если старший (7-й) разряд результата операции равен «1». В противном случае флаг равен «0»
1	Z	**Флаг нуля (Zero)**. Этот флаг устанавливается в «1», если результат выполнения операции равен нулю
0	C	**Флаг переноса (Carry).** Этот флаг устанавливается в «1», если в результате выполнения операции произошел выход за границы байта

GIMSK, TIMSK, GIFR, TIFR (регистры управления прерываниями)

Эти четыре регистра предназначены для управления внешними прерываниями (регистры GIMSK и GIFR) и прерываниями от таймеров (регистры TIMSK и TIFR). Регистры масок GIMSK (General Inter-

rupt MaSK — общий регистр маски прерываний) и TIMSK (Timer Interrupt MaSK — регистр маски прерываний от таймеров) используются для разрешения/запрещения отдельных прерываний, а регистры флагов GIFR (General Interrupt Flag Register — общий регистр флагов прерываний) и TIFR (Timer Interrupt Flag Register — регистр флагов прерываний от таймеров) содержат флаги, показывающие, произошло или нет соответствующее прерывание. В моделях ATtiny28x вместо описанных используются два других регистра: регистр управления прерываниями ICR (Interrupt Control Register) и регистр флагов прерываний IFR (Interrupt Flag Register). Подробно эти регистры будут рассмотрены в разделе 3.5 книги, посвященном прерываниям.

MCUCR (регистр управления микроконтроллером)

Регистр управления микроконтроллером имеется во всех моделях, кроме ATtiny28x, и расположен по адресу $35. Этот регистр содержит ряд флагов, используемых для общего управления микроконтроллером. Состав флагов, размещенных в регистре MCUCR, несколько меняется от модели к модели. Неиспользуемые разряды регистра доступны только для чтения и содержат «0». Все используемые разряды регистра доступны как для чтения, так и для записи в любой момент времени. После сброса микроконтроллера во всех разрядах регистра записаны «0».

Формат этого регистра для различных моделей приведен на **Рис. 1.14**, а описание его разрядов приведено в **Табл. 1.8**. При изменении состояния разрядов ISC01 и ISC00 возможна ложная генерация прерывания INT0. Чтобы этого избежать, рекомендуется на время изменения указанных разрядов запретить внешнее прерывание.

	7	6	5	4	3	2	1	0	
$35	-	-	SE	SM	-	-	ISC01	ISC00	ATtiny11x
Чтение (R)/Запись (W)	R	R	R/W	R/W	R	R	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
$35	-	PUD	SE	SM	-	-	ISC01	ISC00	ATtiny12x
Чтение (R)/Запись (W)	R	R/W	R/W	R/W	R	R	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
$35	-	PUD	SE	SM1	SM0	-	ISC01	ISC00	ATtiny15L
Чтение (R)/Запись (W)	R	R/W	R/W	R/W	R/W	R	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Рис. 1.14. Формат регистра управления микроконтроллером MCUCR

Таблица 1.8. **Разряды регистра MCUCR моделей ATtiny11x/12x/15L**

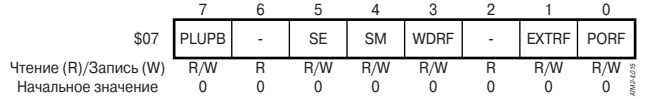
<table>
<tr><th>Разряд</th><th>Название</th><th colspan="3">Описание</th><th>Модель</th></tr>
<tr><td>7</td><td>—</td><td colspan="3">Не используется, читается как «0»</td><td>Все модели</td></tr>
<tr><td rowspan="2">6</td><td>PUD</td><td colspan="3">Запрещение использования внутренних подтягивающих резисторов порта B. Если этот разряд установлен в «1», подключение внутренних подтягивающих резисторов порта B запрещено, если сброшен — разрешено</td><td>ATtiny12x
ATtiny15L</td></tr>
<tr><td>—</td><td colspan="3">Не используется, читается как «0»</td><td>ATtiny11x</td></tr>
<tr><td>5</td><td>SE</td><td colspan="3">Разрешение перехода в режим пониженного энергопотребления. Если этот разряд установлен в «1», то по команде «SLEEP» микроконтроллер переходит в «спящий» режим</td><td>Все модели</td></tr>
<tr><td>4</td><td>SM</td><td colspan="3">Выбор режима пониженного энергопотребления. Состояние этого разряда определяет, в какой режим перейдет микроконтроллер после выполнения команды «SLEEP». Если этот разряд установлен в «1», микроконтроллер переключится в режим «Power Down», если разряд сброшен — в режим «Idle». Более полную информацию см. в разделе 3.3</td><td>ATtiny11x
ATtiny12x</td></tr>
<tr><td>3</td><td>—</td><td colspan="3">Не используется, читается как «0»</td><td>ATtiny11x
ATtiny12x</td></tr>
<tr><td rowspan="6">4, 3</td><td rowspan="6">SM1, SM0</td><td colspan="3">Выбор режима пониженного энергопотребления. Состояние этих разрядов определяет, в какой режим перейдет микроконтроллер после выполнения команды «SLEEP» (см. раздел 4.3)</td><td rowspan="6">ATtiny15L</td></tr>
<tr><td>SM1</td><td>SM0</td><td>Режим</td></tr>
<tr><td>0</td><td>0</td><td>Idle</td></tr>
<tr><td>0</td><td>1</td><td>Режим снижения шумов АЦП</td></tr>
<tr><td>1</td><td>0</td><td>Power Down</td></tr>
<tr><td>1</td><td>1</td><td>Зарезервировано</td></tr>
<tr><td>2</td><td>—</td><td colspan="3">Не используется, читается как «0»</td><td>Все модели</td></tr>
<tr><td rowspan="6">1,0</td><td rowspan="6">ISC01, ISC00</td><td colspan="3">Условие генерации внешнего прерывания INT0</td><td rowspan="5">Все модели</td></tr>
<tr><td>ISC01</td><td>ISC00</td><td>Условие</td></tr>
<tr><td>0</td><td>0</td><td>По НИЗКОМУ уровню на выводе INT0</td></tr>
<tr><td>0</td><td>1</td><td>При любом изменении сигнала на выводе INT0</td></tr>
<tr><td>1</td><td>0</td><td>По спадающему фронту сигнала на выводе INT0</td></tr>
<tr><td>1</td><td>1</td><td>По нарастающему фронту сигнала на выводе INT0</td></tr>
</table>

MCUSR (регистр состояния микроконтроллера)

Регистр состояния микроконтроллера MCUSR, имеющийся во всех моделях, кроме ATtiny28x, расположен по адресу $34. Этот регистр содержит флаги, состояние которых позволяет определить причину, по которой произошел сброс микроконтроллера. Подробно этот регистр будет рассмотрен в разделе 3.4.

MCUCS (регистр управления и состояния микроконтроллера)

Этот регистр присутствует только в моделях ATtiny28x и используется для тех же целей, что и регистры MCUCR и MCUSR в остальных моделях семейства. Формат этого регистра приведен на **Рис. 1.15**, а описание его разрядов — в **Табл. 1.9**.



	7	6	5	4	3	2	1	0
$07	PLUPB	-	SE	SM	WDRF	-	EXTRF	PORF
Чтение (R)/Запись (W)	R/W	R	R/W	R/W	R/W	R	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Рис. 1.15. Формат регистра MCUCS модели ATtiny28x

Таблица 1.9. **Разряды регистра MCUCS моделей ATtiny28x**

Разряд	Название	Описание
7	PLUPB	**Включение внутренних подтягивающих резисторов порта B.** Если этот разряд установлен в «1», внутренние подтягивающие резисторы на всех входах порта B включены, если сброшен — выключены. Если используется любая из альтернативных функций порта B, подтягивающие резисторы отключаются независимо от состояния разряда PLUPB
6	—	Не используется, читается как «0»
5	SE	**Разрешение перехода в режим пониженного энергопотребления**. Если этот разряд установлен в «1», то по команде «SLEEP» микроконтроллер переходит в «спящий» режим
4	SM	**Выбор режима пониженного энергопотребления**. Состояние этого разряда определяет, в какой режим перейдет микроконтроллер после выполнения команды «SLEEP». Если этот разряд установлен в «1», «спящим» режимом является режим «Power Down». Если этот разряд сброшен — режим «Idle». Более полную информацию см. в разделе 3.3
3	WDRF	**Флаг сброса сторожевого таймера**. Этот флаг устанавливается, если сброс микроконтроллера произошел из-за переполнения сторожевого таймера. Разряд сбрасывается в результате сброса по питанию или непосредственной записью в него лог. 0
2	—	Не используется, читается как «0»
1	EXTRF	**Флаг аппаратного сброса**. Этот флаг устанавливается, если произошел аппаратный сброс микроконтроллера (по внешнему сигналу). Разряд сбрасывается в результате сброса по питанию или непосредственной записью в него лог. 0
0	PORF	**Флаг сброса по питанию**. Этот флаг устанавливается в «1» в результате сброса по питанию. Разряд сбрасывается только непосредственной записью в него лог. 0

2.2.2.3. Способы адресации памяти данных

Микроконтроллеры AVR семейства Tiny поддерживают только 4 способа адресации для доступа к различным областям памяти данных. Причем 3 способа из 4-х являются всего лишь разновидностями прямой адресации.

Обратите внимание, что на рисунках этого параграфа, а также далее в книге будет встречаться 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. Микроконтроллеры семейства поддерживают следующие разновидности прямой адресации: прямая адресация одного РОН, прямая адресация двух РОН, прямая адресация РВВ.

Прямая адресация одного регистра общего назначения

Этот способ адресации используется в командах, оперирующих с одним из регистров общего назначения. При этом адрес регистра-операнда (его номер) содержится в разрядах 8...4 (5 бит) слова команды (**Рис. 1.16**). Положение разрядов d на рисунке показано условно.

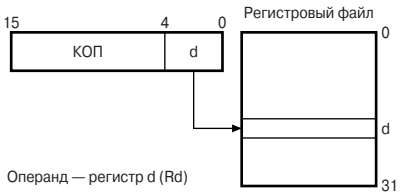


Рис. 1.16. Прямая адресация одного регистра общего назначения

Примером команд, использующих этот способ адресации, являются команды работы со стеком (PUSH, POP), команды инкремента (INC), декремента (DEC), а также некоторые команды арифметических операций.

Прямая адресация двух регистров общего назначения

Этот способ адресации используется в командах, оперирующих одновременно с двумя регистрами общего назначения. При этом адрес регистра-источника содержится в разрядах 9, 3...0 (5 бит), а адрес реги-

стра-приемника в разрядах 8...4 (5 бит) слова команды (**Рис. 1.17**). Положение разрядов r и d на рисунке показано условно.

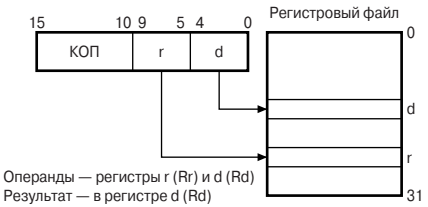


Рис. 1.17. Прямая адресация двух регистров общего назначения

К командам, использующим этот способ адресации, относятся команда пересылки данных из регистра в регистр (MOV), а также большинство команд арифметических операций. Кроме того, этот способ адресации используют даже некоторые команды, имеющие только один регистр-операнд. При этом источником и приемником является один и тот же регистр. В качестве примера можно привести команду очистки регистра (CLR Rd), которая в действительности 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 и регистровым файлом —IN и OUT. В этом случае адрес регистра ввода/вывода содержится в разрядах 10, 9, 3...0 (6 бит), а адрес РОН — в разрядах 8...4 (5 бит) слова команды (**Рис. 1.18**). Положение разрядов r/d и P на рисунке показано условно.

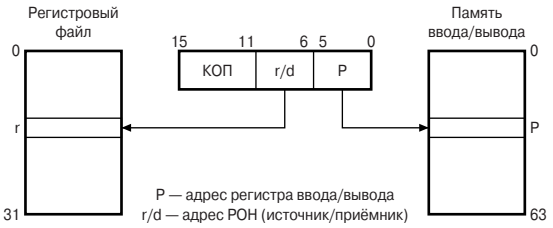


Рис. 1.18. Прямая адресация регистра ввода/вывода

Косвенная адресация

В микроконтроллерах семейства Tiny реализована только простая косвенная адресация. Команды косвенной адресации выполняют обращение к регистру, адрес которого содержится в индексном регистре Z (**Рис. 1.19**). При этом никаких действий с содержимым индексного регистра не производится.

Микроконтроллеры семейства поддерживают 2 команды косвенной адресации: LD Rd, Z (пересылка байта в РОН) и ST Z, Rd (пересылка байта из РОН). Адрес регистра общего назначения содержится в разрядах 8...4 слова команды.

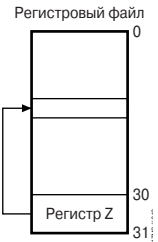


Рис. 1.19. Простая косвенная адресация

2.2.3. Энергонезависимая память данных (EEPROM)

Как уже было сказано, некоторые микроконтроллеры семейства Tiny (ATtiny12x и ATtiny15L) имеют в своем составе энергонезависимую память данных (EEPROM-память). Эта память расположена в собственном адресном пространстве, а ее объем составляет 64 байта.

2.2.3.1. Доступ к EEPROM

Для обращения к EEPROM-памяти используются три регистра ввода/вывода: регистр адреса, регистр данных и регистр управления. Все эти регистры, а также их использование подробно рассматриваются в этом параграфе.

Регистр адреса

Регистр адреса EEPROM-памяти EEAR (EEPROM Address Register) расположен по адресу $1E. В этот регистр загружается адрес ячейки, к которой будет производиться обращение. Регистр адреса доступен как для записи, так и для чтения. Поскольку для адресации 64-х ячеек достаточно 6-разрядного адреса, содержимое двух старших разрядов регистра EEAR игнорируется.

Регистр данных

Регистр данных EEPROM-памяти EEDR (EEPROM Data Register) расположен по адресу $1D. При записи в этот регистр загружаются данные, которые должны быть помещены в EEPROM, а при чтении в этот регистр помещаются данные, считанные из EEPROM.

Регистр управления

Регистр управления EEPROM-памяти EECR (EEPROM Control Register) расположен по адресу $1C. Как следует из названия, данный ре-

гистр используется для управления доступом к EEPROM-памяти. Формат этого регистра показано на **Рис. 1.19**, а его описание приведено в **Табл. 1.10**.

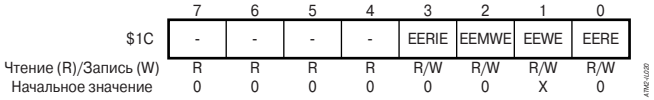


Рис. 1.20. Формат регистра EECR

Таблица 1.10. **Разряды регистра EECR**

Разряд	Название	Описание
7...4	—	Не используются, читаются как «0»
3	EERIE	**Разрешение прерывания от EEPROM**. Данный разряд управляет генерацией прерывания, возникающего при завершении цикла записи в EEPROM. Если этот разряд установлен в «1», прерывания разрешены (если флаг I регистра SREG также установлен в «1»). При сброшенном разряде EEWE (см. далее в таблице) прерывание генерируется постоянно
2	EEMWE	**Управление разрешением записи в EEPROM**. Состояние этого разряда определяет функционирование флага разрешения записи EEWE. Если данный разряд установлен в «1», то при записи в разряд EEWE «1» происходит запись данных в EEPROM. В противном случае установка EEWE в «1» не производит никакого эффекта. После программной установки этот разряд сбрасывается аппаратно через 4 машинных цикла
1	EEWE	**Разрешение записи в EEPROM**. При установке этого разряда в «1» происходит запись данных в EEPROM (если EEMWE равен «1»)
0	EERE	**Разрешение чтения из EEPROM**. После установки этого разряда в «1» выполняется чтение данных из EEPROM. По окончании чтения этот разряд сбрасывается аппаратно

Таким образом, процедура записи одного байта в EEPROM-память состоит из следующих этапов:

- дождаться готовности EEPROM к записи данных (ждать пока не сбросится флаг EEWE (EEPROM Write Enable) регистра EECR);
- загрузить байт данных в регистр EEDR, а требуемый адрес — в регистр EEAR;
- установить в «1» флаг EEMWE (EEPROM Memory Write Enable) регистра EECR;
- в течение 4-х машинных циклов после установки флага EEMWE записать «1» в разряд EEWE регистра EECR.

Длительность цикла записи зависит от частоты внутреннего *RC*-генератора (от значения калибровочной константы) и составляет 3.1...6.8 мс для моделей ATtiny12x и 4.6...8.2 мс для модели ATtiny15L. По окончании цикла записи разряд EEWE аппаратно сбрасывается, после чего программа может начать запись следующего байта. Следует также пом-

нить, что после установки разряда EEWE в «1», процессор пропускает 2 машинных цикла перед выполнением следующей инструкции.

При записи в EEPROM могут возникнуть некоторые проблемы, вызванные прерываниями. При возникновении прерывания между 3-м и 4-м этапами описанной последовательности запись в EEPROM будет сорвана, т. к. за время обработки прерывания флаг EEMWE сбросится в «0». Если в подпрограмме обработки прерывания, возникшего во время записи в EEPROM-память, также происходит обращение к ней, то будет изменено содержимое регистров адреса и данных EEPROM. В результате первая запись (прерванная) будет сорвана.

Для избежания описанных проблем настоятельно рекомендуется запрещать все прерывания (сбрасывать бит I регистра SREG) при выполнении пунктов 2...4 описанной выше последовательности.

С учетом сказанного фрагмент программы, осуществляющий запись в EEPROM, выглядит следующим образом:

```
EEWrite:

            sbic        EECR,EEWE       ;Ждать,пока флаг
            rjmp        EEWrite         ;EEWE не будет сброшен
            cli                         ;Запретить прерывания
            out         EEAR,AddrReg    ;Загрузить адрес
                                        ;(AddrReg — РОН)
            out         EEDR,DataReg    ;Загрузить данные
                                        ;(DataReg — РОН)
            sbi         EECR,EEMWE
            sbi         EECR,EEWE       ;Выдать строб записи
                                        ;в EEPROM
            sei                         ;Разрешить прерывния
                                        ;(если необходимо)
```

Процедура чтения данных из EEPROM гораздо проще, чем процедура записи. После загрузки требуемого адреса в регистр EEAR, программа должна установить разряд EERE регистра EECR в «1». Когда запрошенные данные будут находиться в регистре данных EEDR, произойдет аппаратный сброс этого разряда.

Операция чтения из EEPROM всегда выполняется за один машинный цикл. Кроме того, после установки разряда EERE в «1» процессор пропускает 4 машинных цикла перед началом выполнения следующей инструкции. Поэтому следить в программе за состоянием разряда EERE нет никакой необходимости.

Единственное, на что нужно обратить внимание при чтении из EEPROM, это состояние флага EEWE. Перед выполнением чтения необходимо убедиться, что этот флаг сброшен. В противном случае в

результате загрузки в регистры новых значений адреса и данных во время записи в EEPROM процедура записи будет прервана, а результат этой записи не определен.

С учетом сказанного фрагмент программы, осуществляющий чтение из EEPROM, выглядит следующим образом:

```
EERead:
          sbic     EECR,EEWE           ;Ждать окончания
                                       ;текущей записи,
          rjmp     EERead              ;пока флаг EEWE
                                       ;не станет равным «0»
          out      EEAR, AddrReg       ;Загрузить адрес
                                       ;(AddrReg — РОН)
          sbi      EECR,EERE           ;Выдать строб чтения
                                       ;из EEPROM
          in       DataReg,EEDR        ;Прочитанный байт —
                                       ;в РОН DataReg
```

2.2.3.2. Меры предосторожности

К сожалению, у EEPROM-памяти есть один недостаток: при снижении напряжения питания хранящиеся в ней данные могут быть повреждены. Это может произойти по двум причинам:

- Если напряжение питания ниже некоторой величины, запись в EEPROM будет произведена некорректно.
- Микроконтроллер сам может выполнять команды некорректно, если напряжение питания будет ниже нормы.

Чтобы избежать повреждения данных, хранящихся в EEPROM, достаточно воспользоваться одним из трех следующих решений:

1. Удерживать микроконтроллер в состоянии сброса все время, пока напряжение питания находится ниже нормы. Для этого следует использовать встроенный детектор пониженного напряжения питания (Brown-out Detector, BOD). Более подробно о нем будет рассказано в подразделе 3.4.4.
2. Удерживать микроконтроллер в «спящем» режиме (Power Down) пока напряжение питания находится ниже нормы. Поскольку в этом режиме микроконтроллер не может выполнять никаких команд, такое решение эффективно защищает служебные регистры EEPROM от непреднамеренной записи.
3. Хранить константы во FLASH-памяти программ, если они не должны меняться во время работы программы. Микроконтроллер не может самостоятельно производить запись в FLASH-память. Соответственно, при понижении напряжения питания ее содержимое не будет повреждено.

2.3. Счетчик команд и выполнение программы

2.3.1. Функционирование конвейера

Одной из причин, обуславливающей большое быстродействие микроконтроллеров AVR, является использование двухуровневого конвейера при выполнении программы. Работа этого конвейера показана на **Рис. 1.21**.

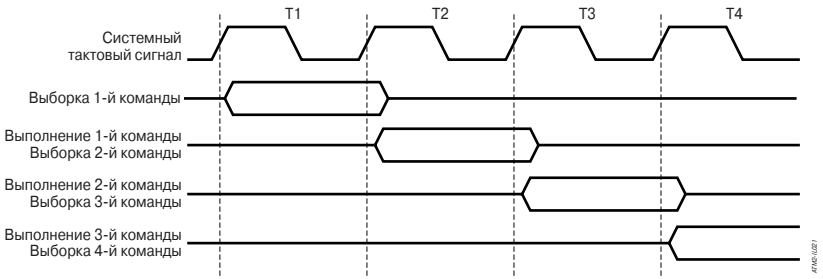


Рис. 1.21. Последовательность выполнения команд в конвейере

Во время первого машинного цикла происходит выборка команды из памяти программ и ее декодирование. Во время второго цикла эта команда выполняется, а параллельно происходит выборка и декодирование второй команды и т. д. В результате фактическое время выполнения каждой команды получается равным одному машинному циклу.

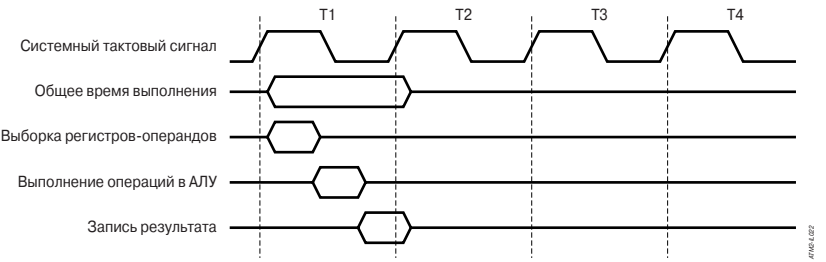


Рис. 1.22. Функционирование АЛУ

Благодаря подключению АЛУ непосредственно к регистровому файлу, оно выполняет одну команду (чтение содержимого двух регистров, выполнение операции и запись результата в регистр-приемник) за один такт (машинный цикл), как показано на **Рис. 1.22**.

2.3.2. Задержки в конвейере

Выше была описана последовательность выполнения команд программы в идеальном случае. Однако в действительности при выполнении некоторых команд может происходить нарушение нормального функционирования конвейера. Наиболее ярким примером команд, вызывающих подобное нарушение, являются команды условного перехода, а также команды типа Test & Skip (проверка и пропуск следующей команды, если результат проверки положительный). В первом случае, если условие, проверяемое командой условного перехода, истинно, выполнение программы будет продолжено с некоторого адреса. А поскольку в конвейере уже произошла выборка команды, расположенной за командой перехода, время выполнения команды перехода увеличивается на один машинный цикл, во время которого происходит выборка команды, расположенной по требуемому адресу.

Во втором случае при выполнении команд типа Test & Skip следующая команда не выполняется в случае истинности проверяемого условия. Однако выборка пропускаемой команды уже произошла. Вследствие того что команда не выполняется, в конвейере образуется «дырка», которая заключается в пропуске одного или двух (в зависимости от пропускаемой команды) машинных циклов. Соответственно команды типа Test & Skip выполняются за один машинный цикл, если результат проверки условия отрицателен, и за два или три цикла, если результат проверки положителен.

Аналогично команды безусловного перехода RJMP (Relative JuMP) и IJMP (Index JuMP), команды вызова подпрограммы RCALL (Relative CALL) и ICALL (Index CALL) и команды возврата из подпрограмм RET (RETurn) и RETI (RETurn Interrupt) также изменяют содержимое счетчика команд PC (Program Counter), вызывая тем самым переход в памяти программ. В результате выполнения этих команд происходит «разрыв» в работе конвейера, а вследствие этого происходит задержка выполнения программы на несколько (2...4) машинных циклов. Для получения более подробной информации обратитесь к описанию команд, приведенному в 3-й части книги.

По той же причине нарушение нормального функционирования конвейера происходит и при возникновении прерывания. Минимальная задержка при этом составляет 4 машинных цикла.

2.3.3. Счетчик команд

Размер счетчика команд составляет 9 (модели ATtiny11x, ATtiny12x, ATtiny15L) или 10 (модели ATtiny28x) разрядов. Напрямую (как регистр) счетчик команд из программы недоступен.

При нормальном выполнении программы содержимое счетчика команд автоматически увеличивается на 1 или на 2 (в зависимости от выполняемой команды) в каждом машинном цикле. Этот порядок нарушается при выполнении команд перехода, вызова и возврата из подпрограмм, а также при возникновении прерываний.

После включения питания, а также после сброса микроконтроллера в счетчик программ автоматически загружается значение $000. Как правило, по этому адресу располагается команда относительного перехода (RJMP) к инициализационной части программы.

При возникновении прерывания в счетчик команд загружается адрес соответствующего вектора прерывания ($001...$010). Если прерывания используются в программе, по этим адресам должны размещаться команды относительного перехода к подпрограммам обработки прерываний. В противном случае основная программа может начинаться непосредственно с адреса $001.

2.3.4. Команды типа «проверка/пропуск» (Test & Skip)

В командах этого типа производится проверка условия, результат которой влияет на выполнение следующей команды. Если условие истинно, следующая команда игнорируется. Например, команда SBRS Rd.b проверяет разряд b регистра Rd и игнорирует следующую команду, если этот разряд равен «1». В действительности переход к следующей инструкции производится увеличением счетчика команд на 1, а пропуск команды требует загрузки нового значения в счетчик команд. Следовательно, когда проверяемое условие истинно, в конвейере возникает задержка. Длительность задержки зависит от размера пропускаемой команды и составляет от одного до двух машинных циклов.

2.3.5. Команды условного перехода

В этих командах производится проверка условия, результат которой влияет на состояние счетчика команд. Если условие истинно, происходит переход по заданному адресу. Если же условие ложно, выполняется следующая команда.

Команды условного перехода имеют ограничение по области действия. В действительности новое значение счетчика команд получается прибавлением к нему или вычитанием из него некоторого смещения. А поскольк-

ку под значение смещения в слове команды отводится всего 7 разрядов, максимальная величина перехода составляет 128 слов (–64...+64).

Так как переход по заданному адресу осуществляется загрузкой нового значения в счетчик команд, то в случае истинности проверяемого условия в конвейере возникает задержка длительностью в один машинный цикл.

2.3.6. Команда безусловного перехода

Для безусловного перехода по требуемому адресу в памяти программ имеется только одна команда — команда относительного перехода RJMP. Процесс выполнения команды заключается в изменении содержимого счетчика команд путем прибавления к нему или вычитания из него некоторого значения, являющегося операндом команды, как показано на **Рис. 1.23.** Положение разрядов k на рисунке показано условно.

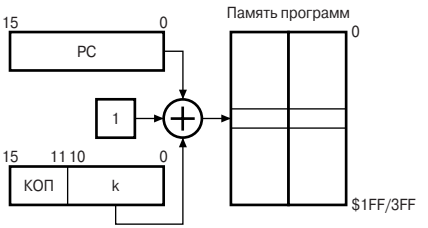


Рис. 1.23. Относительная адресация памяти программ

В программах в качестве операндов этой команды вместо констант используются метки. Ассемблер сам вычисляет величину перехода и подставляет это значение в слово команды. Проиллюстрируем сказанное следующим примером:

```
        cpi r16,$42     ;Сравниваем регистр R16 с числом $42
        brne error      ;Переход, если R16 <> $42
        rjmp ok         ;Безусловный переход
error:
        . . .
ok:     nop             ;Место перехода по команде RJMP
```

Поскольку команда относительного перехода изменяет содержимое счетчика команд, она выполняется за 2 машинных цикла.

2.3.7. Команда вызова подпрограмм

Как и для реализации безусловных переходов, для вызова подпрограмм имеется только одна команда — команда относительного вызова RCALL. Если не принимать во внимание некоторые отличия, описанные ниже, эта команда работает так же, как и команда относительного безусловного перехода RJMP.

Команда RCALL сохраняет в стеке значение счетчика команд. Затем содержимое счетчика команд увеличивается или уменьшается на некоторое значение, являющееся операндом команды (**Рис. 1.23**).

В программах в качестве операндов этой команды, как и в случае команды RJMP, используются метки. Ассемблер сам вычисляет величину перехода и подставляет это значение в слово команды.

Команда относительного вызова подпрограмм выполняется за 3 машинных цикла, 2 из которых затрачиваются на сохранение в стеке двух байт счетчика команд.

2.3.8. Команды возврата из подпрограмм

В конце каждой подпрограммы обязательно должна находиться команда возврата из нее. В системе команд микроконтроллеров семейства имеется две таких команды. Для возврата из обычной подпрограммы, вызываемой командой RCALL, используется команда RET. Для возврата из подпрограммы обработки прерывания используется команда RETI.

Обе команды восстанавливают из стека содержимое счетчика команд, сохраненное там перед переходом к подпрограмме. Команда возврата из подпрограммы RETI дополнительно устанавливает в «1» флаг общего разрешения прерываний I регистра SREG, сбрасываемый аппаратно при возникновении прерывания.

На выполнение каждой из команд возврата из подпрограммы требуется 4 машинных цикла.

2.4. Стек

В микроконтроллерах AVR семейства Tiny стек реализован аппаратно. Глубина стека равна трем уровням, а размер равен размеру счетчика команд (9 или 10 разрядов). Стек расположен в собственной области памяти и имеет LIFO-организацию (Last In First Out — последним вошел, первым вышел), т. е. значение, записанное последним, будет прочитано первым.

При вызове подпрограммы адрес команды, расположенной за командой RCALL, сохраняется в стеке. При возврате из подпрограммы этот адрес извлекается из стека и загружается в счетчик команд. Тоже происходит и во время прерывания. При генерации прерывания адрес следующей команды сохраняется в стеке, а при возврате из подпрограммы обработки прерывания он восстанавливается из стека.

Непосредственно из программы стек недоступен, т. к. в наборе команд микроконтроллера отсутствуют команды занесения в стек и извлечения из стека. Указатель стека также недоступен из программы, т. е. он не может быть явно прочитан или модифицирован. Поэтому микрокон-

троллер сам управляет перемещением данных по стеку. Чтобы лучше понять работу стека, обратитесь к **Рис. 1.24** и **Рис. 1.25**.

Рассмотрим выполнение команды RCALL (**Рис. 1.24**): содержимое счетчика команд пересылается на 1-й уровень стека, а предыдущие значения предварительно «сползают» на один уровень (значение, находившееся на первом уровне, перемещается на второй уровень и т. д.). Заметим, что в результате этой операции будет потеряно значение, расположенное на 3-м уровне стека.

При выполнении команды возврата из подпрограммы RET или RETI (**Рис. 1.25**) значение, хранящееся на 1-м уровне стека, заносится в счетчик команд. Во время этой операции все значения «поднимаются» на один уровень вверх (значение, находившееся на втором уровне, перемещается на первый уровень и т. д.).

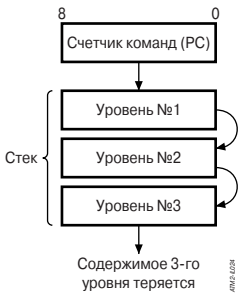


Рис. 1.24. Работа стека при выполнении команды RCALL

Рис. 1.25. Работа стека при выполнении команды RET

Глава 3. **Устройство управления микроконтроллеров семейства Tiny**

3.1. Общие сведения

Микроконтроллеры семейства Tiny предоставляют пользователю широкие возможности по выбору источника тактового сигнала и его частоты. Здесь может быть использован встроенный генератор с внутренним или внешним кварцевым резонатором, внешний сигнал синхронизации, встроенный *RC*-генератор с внутренней или внешней времязадающей *RC*-цепочкой. Исключение составляет лишь микроконтроллер ATtiny15L — в этой модели тактовый сигнал может вырабатываться только встроенным генератором с внутренней *RC*-цепочкой.

Все микроконтроллеры семейства Tiny имеют два режима пониженного энергопотребления: Idle (режим ожидания) и Power Down. А микроконтроллер ATtiny15L имеет еще и дополнительный режим — ADC Noise Reduction (режим снижения шумов АЦП). Эти режимы часто называют одним словом — «спящий» режим. Каждый из этих режимов позволяет значительно сократить энергопотребление в периоды бездействия микроконтроллера. Вход в этот режим выполняется по команде SLEEP. При выходе микроконтроллера из «спящего» режима производится его реинициализация (сброс в исходное состояние, далее просто сброс).

Разумеется, сброс микроконтроллера может произойти не только при его «пробуждении». Другими событиями, при которых осуществляется сброс микроконтроллера, являются подача напряжения питания, снижение напряжения питания ниже минимально допустимого уровня, срабатывание сторожевого таймера, появление на выводе $\overline{\text{RESET}}$ сигнала НИЗКОГО уровня.

В качестве источников прерывания в микроконтроллерах семейства могут выступать периферийные устройства, а также сигналы на некоторых выводах (входы внешних прерываний). Так как состав периферийных устройств зависит от модели, то и число источников прерываний в каждой модели различно.

3.2. Тактовый генератор

Все микроконтроллеры семейства Tiny, за исключением модели ATtiny15L, могут работать с встроенным генератором с внутренней или внешней времязадающей *RC*-цепочкой, с внешним керамическим либо кварцевым резонатором или, наконец, от сигнала внешней синхронизации. Выбор режима работы осуществляется программированием конфигурационных ячеек (FUSE Bits, FUnction SElect Bits) CKSEL3...0 (ClocK SELect) (CKSEL2...0 для ATtiny11x). Требуемые значения для каждого режима работы приведены в **Табл. 1.11**. Более подробно о конфигурационных ячейках и слове конфигурации будет рассказано в четвертой части книги, посвященной программированию микроконтроллеров.

Таблица 1.11. **Выбор режима работы тактового генератора**

Режим работы	CKSEL2...0 (ATtiny11x)	CKSEL3..0 (ATtiny12x, ATtiny28x)
Внешний кварц или керамический резонатор	111	1111...1010
Внешний низкочастотный резонатор	110	1001...1000
Внешняя времязадающая *RC*-цепочка	101	0111...0101
Внутренняя времязадающая *RC*-цепочка*	100	0100...0010
Внешний сигнал синхронизации	000	0001...0000
Зарезервировано	Прочие	—
* Режим по умолчанию.		

От значения, занесенного в эти ячейки, зависит также длительность задержки сброса t_{TOUT} (см. раздел 3.4). Напоминаем еще раз, что микроконтроллер ATtiny15L может работать только от внутреннего *RC*-генератора.

3.2.1. Кварцевый генератор

Резонатор подключается к выводам XTAL1 и XTAL2 микроконтроллеров, как показано на **Рис. 1.26**. Эти выводы являются соответственно входом и выходом инвертора встроенного тактового генератора. При необходимости тактовый сигнал микроконтроллера можно использовать для управления какими-либо внешними устройствами. Этот сигнал снимается с вывода XTAL2, при этом между выводом и внешней схемой обязательно должен быть буфер (**Рис. 1.26**).

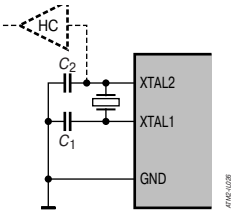


Рис. 1.26. Подключение кварцевого или керамического резонатора

Емкости конденсаторов C_1 и C_2, подключаемых между выводами резонатора и общим проводом, зависят от частоты и типа резонатора и приводятся в документации на него. Обратите внимание, что в моделях ATtiny28x уже имеются внутренние конденсаторы емкостью ~50 пФ. При программировании конфигурационной ячейки INTCAP (записи в нее лог. 0) эти конденсаторы подключаются между выводами XTAL1/XTAL2 и общим проводом.

3.2.2. Внешний сигнал синхронизации

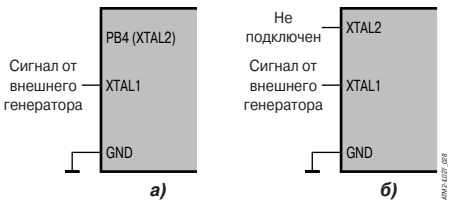


Рис. 1.27. Подключение внешнего источника тактового сигнала:
а — ATtiny11x/12x; *б* — ATtiny28x

Сигнал от внешнего источника подается на вывод XTAL1, как показано на **Рис. 1.27**. Этот сигнал должен удовлетворять требованиям микроконтроллера по частоте, скважности и уровням напряжения. В моделях ATtiny11x и ATtiny12x вывод XTAL2 в этом случае может работать как контакт ввода/вывода. В моделях ATtiny28x вывод XTAL2 оставляют неподключенным.

3.2.3. Встроенный генератор с внешней или внутренней *RC*-цепочкой

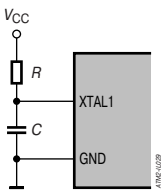


Рис. 1.28. Подключение внешней *RC*-цепочки

При реализации приложений, не требующих высокой временной точности, можно использовать внешний *RC*-генератор. При этом *RC*-цепочка подключается к выводу XTAL1, как показано на **Рис. 1.28**.

Частота тактового сигнала определяется значением сопротивления *R* и емкости *C*. Рекомендуется использовать резистор с сопротивлением 3.3...100 кОм и конденсатор емкостью не менее 20 пФ. Значения *R* и *C* для «типовых» частот приведены в **Табл. 1.12**.

Таблица 1.12. **Параметры внешней *RC*-цепочки для «типовых» частот**

R [кОм]	C [пФ]	f [кГц]
100	70	100
31.5	20	1000
6.5	20	4000

Использование встроенного генератора с внутренней *RC*-цепочкой является наиболее экономичным решением, т. к. при этом не требуются внешние компоненты и не задействуются контакты ввода/вывода. Вместе с тем это решение пригодно только для приложений, не требующих высокой временной точности.

Номинальные частоты встроенного *RC*-генератора для различных моделей приведены в **Табл. 1.13**.

Таблица 1.13. **Номинальные частоты встроенного *RC*-генератора**

Модель	Частота [кГц]
ATtiny11x	1100
ATtiny12x	1200
ATtiny15L	1600
ATtiny28x	1200

Все микроконтроллеры за исключением моделей ATtiny11x позволяют подстраивать (калибровать) частоту генератора. Дело в том, что действительная частота генератора, получаемая при изготовлении микросхемы, может отличаться от экземпляра к экземпляру. Отсюда и возникает необходимость в подстройке генератора для достижения номинальной частоты.

Для подстройки генератора предназначен регистр OSCCAL, расположенный по адресу $31 в моделях ATtiny12x, ATtiny15L и по адресу $00 для моделей ATtiny28x. Чем больше значение, записанное в этом регистре, тем больше частота генератора.

Значение, необходимое для подстройки генератора на номинальную частоту (с точностью ±1%), записывается при изготовлении микроконтроллера в специальную калибровочную ячейку. Содержимое этой ячейки доступно только в режиме программирования микроконтроллеров. Соответственно программатор должен считать содержимое этой ячейки и записать его в заранее определенное место FLASH-памяти программ (как правило, в последнюю ячейку). А в самом начале программы следует прочитать содержимое по этому адресу и записать его в регистр OSCCAL.

Следует помнить, что встроенный генератор предназначен для работы на номинальной частоте. Поэтому подстройка на другие частоты

хотя и возможна, но не гарантируется. Более того, в микроконтроллерах ATtiny12x встроенный *RC*-генератор также определяет временные параметры доступа к EEPROM-памяти. Поэтому увеличение частоты генератора более чем на 10% может привести к невозможности записи в EEPROM.

3.3. Режимы пониженного энергопотребления

Все модели микроконтроллеров семейства поддерживают два различных «спящих» режима: Idle (ждущий режим) и Power Down (режим микропотребления). А микроконтроллер ATtiny15L поддерживает еще и третий режим — ADC Noise Reduction (режим снижения шумов АЦП).

Переключение в любой из режимов пониженного энергопотребления осуществляется командой SLEEP. При этом флаг SE (Sleep Enable – спящий режим разрешен) регистра управления MCUCR должен быть установлен в «1». Чтобы избежать непреднамеренного переключения микроконтроллера в «спящий» режим, рекомендуется устанавливать этот флаг непосредственно перед выполнением команды SLEEP. Выбор конкретного режима определяется состоянием флагов SM1 и SM0 (Sleep Mode) для модели ATtiny15L или флага SM для остальных: «1» или «0». Эти флаги также расположены в регистре MCUCR. Соответствие между состоянием этих флагов и режимом пониженного энергопотребления приведено в **Табл. 1.14**.

Таблица 1.14. **Выбор режима пониженного энергопотребления**

ATtiny15L		Прочие	Режим
SM1	SM0	SM	
0	0	0	Idle
0	1	—	ADC Noise Reduction
1	0	1	Power Down
1	1	—	Зарезервировано

Выход из «спящего» режима может быть осуществлен в результате прерыванияили сброса. В первом случае микроконтроллер переходит в рабочий режим, останавливается на 4 машинных цикла, выполняет подпрограмму обработки прерывания, после чего выполнение программы возобновляется с инструкции, следующей за командой SLEEP. Содержимое РОН, ОЗУ и РВВ при этом не изменяется. При сбросе микроконтроллер переходит в рабочий режим, и выполнение программы начинается с адреса $000. Более подробно процесс сброса будет рассмотрен в разделе 3.4.

3.3.1. Режим Idle

В этом режиме прекращает работу ЦПУ микроконтроллера, а все остальные периферийные устройства (таймеры/счетчики, аналоговый компаратор, АЦП, сторожевой таймер), а также подсистема прерываний продолжают функционировать. За счет этого выход из режима Idle возможен как по внешнему прерыванию, так и по внутреннему, например при переполнении таймера. Для еще большего уменьшения энергопотребления в режиме Idle рекомендуется отключать встроенный аналоговый компаратор, если не требуется выход из «спящего» режима по прерыванию от компаратора.

Основным преимуществом режима Idle является быстрая реакция на события, «пробуждающие» микроконтроллер. Другими словами, выполнение программы начинается сразу же после перехода из режима Idle в рабочий режим.

3.3.2. Режим Power Down

В режиме Power Down функционирование всех систем микроконтроллера, включая тактовый генератор, прекращается. Единственными узлами, продолжающими работать в этом режиме, являются сторожевой таймер (если он включен) и подсистема обработки внешних прерываний. Соответственно выход из режима Power Down возможен только при возникновении одного из четырех событий:

- внешний (аппаратный) сброс;
- сброс от сторожевого таймера;
- генерация внешнего прерывания по уровню;
- генерация прерывания из-за изменения состояния вывода.

Чтобы микроконтроллер мог выйти из режима Power Down, по двум последним прерываниям должны выполняться следующие условия:

- ATtiny11x: длительность активного сигнала должна превышать величину задержки сброса t_{TOUT} (см. раздел 3.4). Под активным здесь понимается сигнал, появление которого привело к генерации прерывания;
- ATtiny15L: длительность активного сигнала должна быть не меньше двух периодов сигнала тактового генератора сторожевого таймера (5.8 мкс при V_{CC} = 3 В);
- ATtiny12x, ATtiny28x: активный сигнал на выводе микроконтроллера должен удерживаться в течение меньшего из двух интервалов — удвоенного периода тактового генератора сторожевого таймера (5.8 мкс при V_{CC} = 3 В) и периода восстановления, определяемого содержимым конфигурационных ячеек CKSEL3...0 (**Табл. 1.15**).

Таблица 1.15. **Задание периода восстановления**

CKSEL3...0	Период восстановления
1111...1101	1K CK*
1100...1010	16K CK
1001	1K CK
1000	32K CK
0111...0000	16 CK
* CK — период тактового сигнала микроконтроллера.	

Следует помнить, что между наступлением события, приводящего к выходу микроконтроллера из режима Power Down, и началом работы микроконтроллера проходит некоторое время, в течение которого тактовый генератор микроконтроллера выходит на рабочий режим.

3.3.3. Режим ADC Noise Reduction

Данный режим имеется только в микроконтроллере ATtiny15L, поскольку это единственная модель семейства, имеющая в своем составе АЦП. В этом режиме прекращает работу ЦПУ микроконтроллера, а АЦП, подсистема обработки внешних прерываний, сторожевой таймер и тактовый генератор продолжают функционировать. За счет этого уменьшаются помехи на входах АЦП, вызываемые работой систем микроконтроллера, что, в свою очередь, позволяет повысить точность преобразования. Если АЦП включен, сразу же после перехода в «спящий» режим автоматически начинается преобразование.

Возврат микроконтроллера в рабочий режим может произойти только при возникновении любого из следующих событий:

- аппаратный сброс;
- сброс от сторожевого таймера;
- генерация внешнего прерывания по уровню;
- генерация прерывания из-за изменения состояния вывода;
- генерация прерывания от АЦП.

3.4. Сброс

Сброс (реинициализация) микроконтроллера переводит его в исходное устойчивое состояние. Сброс может быть вызван следующими событиями:

- включение напряжения питания микроконтроллера;
- падение напряжения питания ниже заданной величины (только для моделей ATtiny12x и ATtiny15L);
- тайм-аут сторожевого таймера;
- подача сигнала НИЗКОГО уровня на вывод $\overline{\text{RESET}}$ (сброс).

При наступлении любого из перечисленных событий во все регистры ввода/вывода заносятся их начальные значения (см. описания конкретных регистров), а в счетчик команд загружается значение $000 (адрес вектора сброса). Если в программе используются какие-либо прерывания, то по этому адресу должна находиться команда относительного перехода RJMP на начало программы (к ее инициализационной части). Если же прерывания в программе не используются, то программа может начинаться непосредственно с адреса $000.

Следует сразу же сказать, что разные модели микроконтроллеров семейства предоставляют различные возможности по управлению процессом сброса. Однако логика работы схемы сброса одинакова для всех моделей.

При наступлении какого-либо из поддерживаемых событий формируется внутренний сигнал сброса. Одновременно запускается таймер формирования задержки сброса. По истечении определенного промежутка времени внутренний сигнал сброса снимается и начинается выполнение программы.

Все микроконтроллеры семейства позволяют определить событие, в результате которого произошел сброс устройства. Для этой цели в моделях ATtiny11x, ATtiny12x и ATtiny15L используется регистр состояния микроконтроллера MCUSR, расположенный по адресу $34. Формат этого регистра различается для каждой группы микроконтроллеров семейства. В моделях ATtiny28x для определения источника сброса используется регистр управления и состояния микроконтроллера MCUCS, расположенный по адресу $07. Оба указанных регистра содержат набор флагов, состояние которых зависит от события, вызвавшего сброс устройства.

Структурные схемы подсистемы сброса будут приведены далее при обсуждении особенностей, присущих конкретным моделям.

3.4.1. Сброс по включению питания

Все микроконтроллеры семейства Tiny имеют в своем составе схему сброса при включении питания (схема POR, Power-on Reset). Эта схема удерживает микроконтроллер в состоянии сброса до тех пор, пока нарастающее напряжение питания не достигнет некоторого порогового значения. Работает эта схема следующим образом.

Когда нарастающее напряжение питания достигает порогового значения V_{POT} (Power-On Threshold, порог включения), начинает работать таймер задержки сброса. По окончании счета (после формирования задержки t_{TOUT}) внутренний сигнал сброса снимается и происходит запуск микроконтроллера.

Существует два способа управления состоянием вывода $\overline{\text{RESET}}$. Если время нарастания напряжения источника питания до порогового значения известно и не превышает величины t_{TOUT} для конкретной схемы, можно использовать первый способ, при котором напряжение на выводе $\overline{\text{RESET}}$ «повторяет» напряжение питания. Соответствующие данному способу временные диаграммы показаны на **Рис. 1.29**. Для реализации этого способа вывод $\overline{\text{RESET}}$ нужно подключить к источнику питания либо (для ATtiny28x) оставить неподключенным, т. к. в этих моделях вывод сброса уже подтянут (подключен) к шине питания V_{CC} внутренним резистором сопротивлением 100...500 кОм.

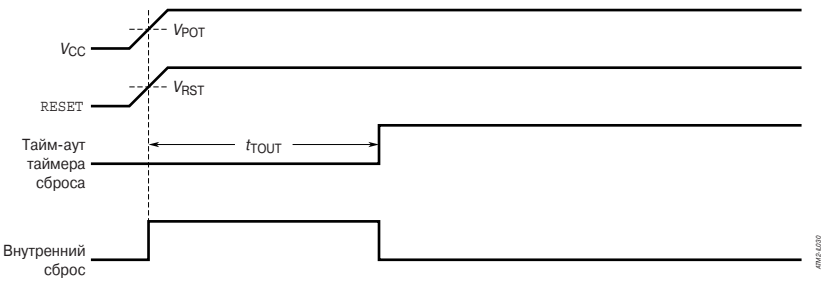


Рис. 1.29. Временные диаграммы сигналов при сбросе по включению питания; вывод $\overline{\text{RESET}}$ подключен к V_{CC}

Второй способ управления состоянием вывода $\overline{\text{RESET}}$ является классическим: сигнал ВЫСОКОГО уровня подается на вывод только после установления напряжения питания. Временные диаграммы, соответствующие этому способу, показаны на **Рис. 1.30**.

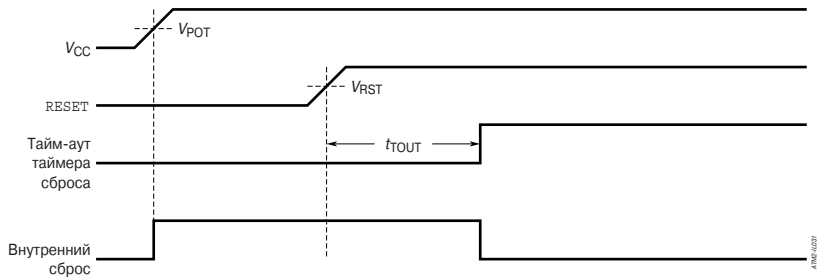


Рис. 1.30. Временные диаграммы сигналов (сброс при включении питания, вывод $\overline{\text{RESET}}$ управляется внешней схемой)

В этом случае работой таймера задержки сброса будет управлять схема аппаратного сброса, о которой будет рассказано чуть позже. Соответственно этот таймер начинает работать при достижении напряжения на выводе $\overline{\text{RESET}}$ порогового значения V_{RST}.

Данное решение является более дорогостоящим, т. к. требует применения внешних компонентов. С другой стороны, при использовании этого способа время запуска устройства будет достаточно большим, что гарантирует надежный и корректный запуск микроконтроллера в схеме.

3.4.2. Аппаратный сброс

Аппаратный (или внешний) сброс микроконтроллера реализуется подачей на вывод $\overline{\text{RESET}}$ сигнала НИЗКОГО уровня. Микроконтроллер остается в состоянии сброса до тех пор, пока на выводе $\overline{\text{RESET}}$ будет присутствовать сигнал НИЗКОГО уровня. Длительность импульса сброса должна быть не менее 500 нс, в противном случае сброс микроконтроллера не гарантируется. При достижении напряжением на выводе $\overline{\text{RESET}}$ порогового значения V_{RST} запускается таймер задержки сброса. По окончании счета (после формирования задержки t_{TOUT}) внутренний сигнал сброса снимается и происходит запуск микроконтроллера. Последовательность сигналов при аппаратном сбросе показана на **Рис. 1.31**.

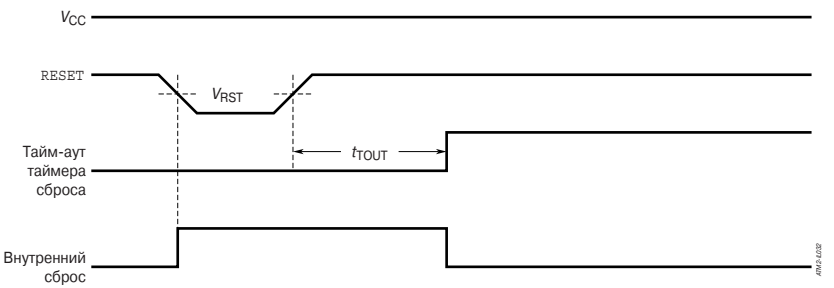


Рис. 1.31. Временные диаграммы сигналов при аппаратном сбросе

3.4.3. Сброс от сторожевого таймера

По тайм-ауту сторожевого таймера (если он включен) генерируется короткий положительный импульс сброса, длительность которого равна одному периоду тактового сигнала микроконтроллера. По спадающему фронту этого импульса запускается таймер задержки сброса. По

окончании счета (после формирования задержки t_{TOUT}) внутренний сигнал сброса снимается и происходит запуск микроконтроллера. Временные диаграммы, соответствующие этому событию, показаны на **Рис. 1.32**.

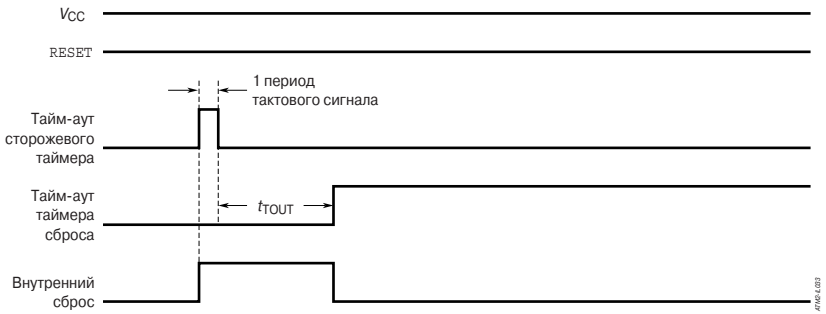


Рис. 1.32. Временные диаграммы сигналов при сбросе от сторожевого таймера

3.4.4. Сброс при снижении напряжения питания

Модели микроконтроллеров ATtiny12x и ATtiny15L имеют в своем составе схему BOD (Brown-Out Detection), которая отслеживает напряжение источника питания. Если работа этой схема разрешена, то при снижении напряжения питания ниже некоторого значения она переводит микроконтроллер в состояние сброса. Когда напряжение питания вновь увеличится до порогового значения, запускается таймер задержки сброса. После формирования задержки t_{TOUT} внутренний сигнал сброса снимается и происходит запуск микроконтроллера. Временные диаграммы, соответствующие данному виду сброса, показаны на **Рис. 1.33**.

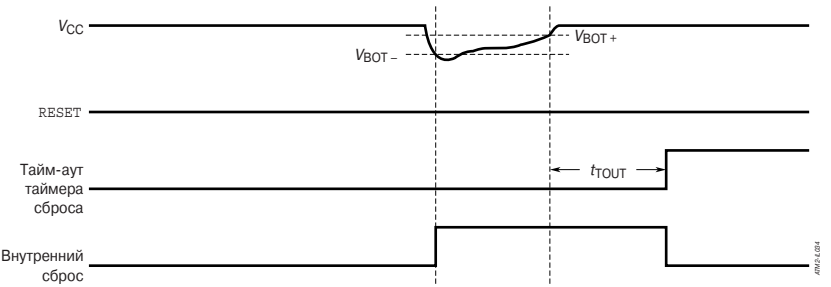


Рис. 1.33. Временные диаграммы сигналов (сброс при снижении напряжения питания)

Работой схемы BOD управляет конфигурационная ячейка BODEN (BOD Enable — режим BOD разрешен), расположенная в выделенной области FLASH-памяти. Для разрешения работы схемы в этой ячейке должен быть записан «0». Порог срабатывания (V_{BOT}) определяется состоянием конфигурационной ячейки BODLEVEL, которая расположена в той же области. Если в этой ячейке записана «1», порог срабатывания равен 1.8 В для ATtiny12x и 2.7 В для ATtiny15L. Если же в ней записан «0» (после ее программирования), порог срабатывания равен 2.7 В и 4.0 В для моделей ATtiny28x и ATtiny15L соответственно. Для уменьшения вероятности ложных срабатываний порог переключения схемы имеет гистерезис, равный 50 мВ. Переход в режим сброса начинается при снижении напряжения питания до нижнего порога V_{BOT-}, а выход из этого режима — при достижении верхнего порога V_{BOT+}. Кроме того, срабатывание схемы BOD произойдет только в том случае, если длительность провала напряжения питания превысит 3 мкс для V_{BOT} = 4.0 В, 7 мкс для V_{BOT} = 2.7 В или 24 мкс для V_{BOT} = 1.8 В.

3.4.5. Управление схемой сброса

В данном подразделе рассмотрены возможности микроконтроллеров семейства Tiny по управлению схемой сброса.

ATtiny11x

Структурная схема подсистемы сброса микроконтроллеров ATtiny11x приведена на **Рис. 1.34**. Как видно из рисунка, таймер задержки сброса работает от *RC*-генератора сторожевого таймера, независимого от основного тактового генератора микроконтроллера.

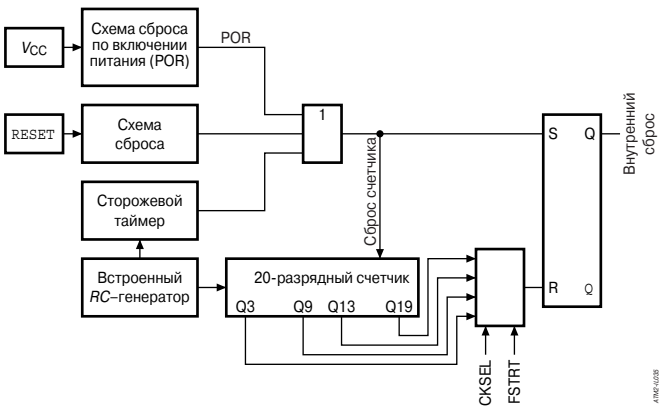


Рис. 1.34. Структурная схема подсистемы сброса микроконтроллеров ATtiny11x

В этих моделях длительность задержки сброса t_{TOUT} зависит от состояния конфигурационной ячейки FSTRT (Function STaRT) и от режима работы тактового генератора (от состояния конфигурационных ячеек CKSEL2...CKSEL0). Соответствующие значения задержек для V_{CC} = 2.7 В (для других значений V_{CC} задержки будут другими) приведены в **Табл. 1.16**.

Таблица 1.16. **Задержка сброса t_{TOUT} в микроконтроллерах ATtiny11x**

Режим работы тактового генератора	CKSEL2...0	t_{TOUT} (V_{CC} = 2.7 В)	
		FSTRT = «1»	FSTRT = «0»
Внешний кварц или керамический резонатор	111	67 мс	4.2 мс
Внешний низкочастотный резонатор	110	4.2 с	4.2 с
Внешняя *RC*-цепочка	101	4.2 мс	67 мкс
Внутренняя *RC*-цепочка	100	4.2 мс	67 мкс
Внешний сигнал синхронизации	000	4.2 мс	5 тактов при сбросе, 2 такта при выходе из режима Power Down

Максимальная величина порогового напряжения для вывода $\overline{\text{RESET}}$ V_{RST} составляет $0.6V_{CC}$. Типовое значение пороговой величины напряжения питания V_{POT} составляет 1.4 В.

Формат регистра состояния MCUSR микроконтроллеров ATtiny11x показан на **Рис. 1.35**.

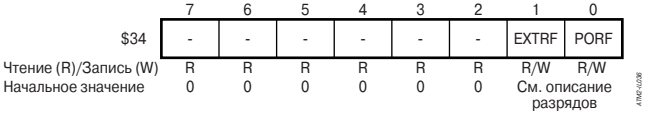


Рис. 1.35. Формат регистра MCUSR моделей ATtiny11x

Как видно из рисунка, в этом регистре задействовано только два младших разряда. Флаг PORF (POwer-on Reset Flag, 0-й разряд) устанавливается в «1» в результате сброса по питанию. При всех остальных событиях, вызывающих сброс микроконтроллера, состояние этого флага не меняется. А флаг EXTRF (EXTernal Reset Flag, 1-й разряд) устанавливается в «1» после аппаратного сброса. После сброса от сторожевого таймера состояние этого флага не меняется, а после сброса по питанию его состояние не определено.

Соответственно, рекомендуется в самом начале программы сбросить оба разряда регистра. Если затем во время работы программы произойдет сброс устройства, источник сброса можно будет определить согласно **Табл. 1.17**.

Таблица 1.17. **Определение источника сброса в микроконтроллерах ATtiny11x**

PORF	EXTRF	Источник сброса
0	0	Сброс от сторожевого таймера
0	1	Аппаратный сброс
1	0	Сброс по питанию
1	1	Сброс по питанию

ATtiny12x

Структурная схема подсистемы сброса микроконтроллеров моделей ATtiny12x приведена на **Рис. 1.36**.

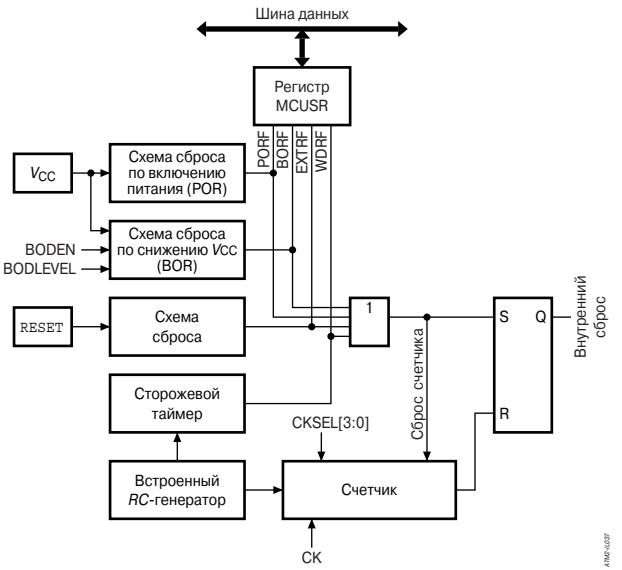


Рис. 1.36. Структурная схема подсистемы сброса микроконтроллеров ATtiny12x

В этих моделях длительность задержки сброса t_{TOUT} зависит от режима работы тактового генератора (от состояния конфигурационных ячеек CKSEL2...CKSEL0) и от состояния конфигурационной ячейки BODLEVEL. Соответствующие значения задержек приведены в **Табл. 1.18**.

Таблица 1.18. **Задержка сброса t_{TOUT} в микроконтроллерах ATtiny12x**

Режим работы тактового генератора	CKSEL3...0	t_{TOUT}	
		V_{CC} = 1.8 В BODLEVEL = «1»	V_{CC} = 2.7 В BODLEVEL = «0»
Внешний кварц или керамический резонатор	1111	1K CK*	1K CK
	1110	3.6 мс + 1K CK	4.2 мс + 1K CK
	1101	57 мс + 1K CK	67 мс + 1K CK
	1100	16K CK	16K CK
	1011	3.6 мс + 16K CK	4.2 мс + 16K CK
	1010	57 мс + 16K CK	67 мс + 16K CK
Внешний низкочастотный резонатор	1001	57 мс + 1K CK	67 мс + 1K CK
	1000	57 мс + 32K CK	67 мс + 32K CK
Внешняя *RC*-цепочка	0111	6 CK	6 CK
	0110	3.6 мс + 6 CK	4.2 мс + 6 CK
	0101	57 мс + 6 CK	67 мс + 6 CK
Внутренняя *RC*-цепочка	0100	6 CK	6 CK
	0011	3.6 мс + 6 CK	4.2 мс + 6 CK
	0010**	57 мс + 6 CK	67 мс + 6 CK
Внешний сигнал синхронизации	0001	6 CK	6 CK
	0000	3.6 мс + 6 CK	4.2 мс + 6 CK

* CK — период тактового сигнала микроконтроллера.

** Режим по умолчанию.

Для формирования первой части задержки используется *RC*-генератор сторожевого таймера (**Рис. 1.36**). Задержке в 3.6 мс соответствуют 256 тактов сигнала генератора, 4.2 мс — 4096 (4K), 57 мс — 1024 (1K), 67 мс — 16384 (16K). А поскольку частота *RC*-генератора зависит от напряжения питания микроконтроллера, то и величина задержки будет зависеть от напряжения питания.

Максимальная величина порогового напряжения для вывода $\overline{\text{RESET}}$ V_{RST} составляет $0.6V_{CC}$. Типовое значение пороговой величины напряжения питания V_{POT} составляет 1.4 В, если схема BOD выключена, и 1.2 В, если выключена.

Формат регистра состояния MCUSR микроконтроллеров ATtiny12x показан на **Рис. 1.37**, а назначение его разрядов приведено в **Табл. 1.19**.

	7	6	5	4	3	2	1	0
$34	-	-	-	-	WDRF	BORF	EXTRF	PORF
Чтение (R)/Запись (W)	R	R	R	R	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	См. описание разрядов			

ATMEL 0208

Рис. 1.37. Формат регистра MCUSR в микроконтроллерах ATtiny12x и ATtiny15L

Таблица 1.19. **Разряды регистра MCUSR моделей ATtiny12x и ATtiny15L**

Разряд	Название	Описание
7...4	—	Не используются, читаются как «0»
3	WDRF	**Флаг сброса от сторожевого таймера.** Устанавливается в «1», если источником сброса был сторожевой таймер. Разряд сбрасывается в результате сброса по питанию или непосредственной записью в него «0»
2	BORF	**Флаг сброса по снижению питания.** Устанавливается в «1», если источником сброса была подсистема BOD. Разряд сбрасывается в результате сброса по питанию или непосредственной записью в него «0»
1	EXTRF	**Флаг аппаратного сброса**. Устанавливается в «1», если сброс произошел в результате подачи на вывод сброса сигнала НИЗКОГО уровня. Разряд сбрасывается в результате сброса по питанию или непосредственной записью в него «0»
0	PORF	**Флаг сброса при включении питания.** Устанавливается в «1» после подачи напряжения питания на микроконтроллер. Разряд сбрасывается только непосредственной записью в него «0»

ATtiny15L

Структурная схема подсистемы сброса микроконтроллера ATtiny15L приведена на **Рис. 1.38**.

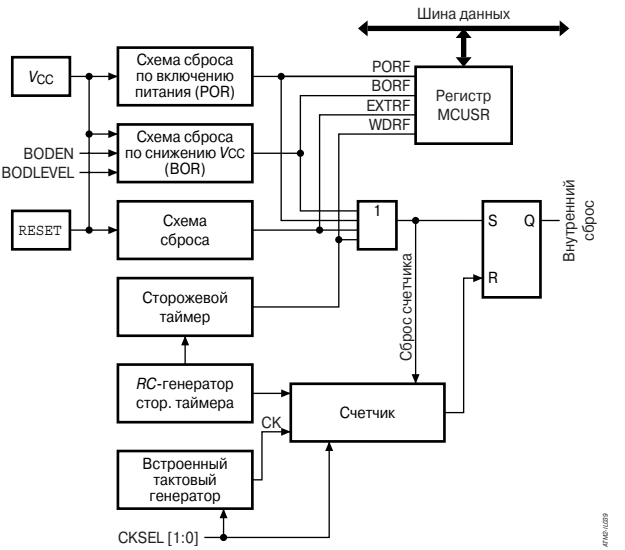


Рис. 1.38. Структурная схема подсистемы сброса микроконтроллера ATtiny15L

В этих моделях длительность задержки сброса t_{TOUT} определяется состоянием конфигурационных ячеек CKSEL1 и CKSEL0, а также состоянием схемы BOD (конфигурационная ячейка BODEN). Значения задержек, соответствующие различным комбинациям указанных конфигурационных ячеек, приведены в **Табл. 1.20**.

Таблица 1.20. **Задержка сброса t_{TOUT} в микроконтроллере ATtiny15L**

BODEN	CKSEL1....0	t_{TOUT}	
		V_{CC} = 2.7 В	V_{CC} = 5.0 В
X	00*	256 мс + 18 CK	64 мс + 18 CK
X	01	256 мс + 18 CK	64 мс + 18 CK
X	10	16 мс + 18 CK	4 мс + 18 CK
1	11	32 мкс + 18 CK	8 мкс + 18 CK
0	11	128 мкс + 18 CK	32 мкс + 18 CK
* Значение по умолчанию.			

Для формирования первой части задержки используется *RC*-генератор сторожевого таймера. Соответствие между числом тактов сигнала генератора и реализуемой задержкой приведено в **Табл. 1.21**.

Таблица 1.21. **Число тактов сигнала генератора сторожевого таймера**

V_{CC} [В]	Задержка	Число тактов
2.7	32 мкс	8
2.7	128 мкс	32
2.7	16 мс	4K
2.7	256 мс	64K
5.0	8 мкс	8
5.0	32 мкс	32
5.0	4 мс	4K
5.0	64 мс	64K

Поскольку частота *RC*-генератора сторожевого таймера зависит от напряжения питания микроконтроллера, действительные значения задержек могут отличаться от приведенных в **Табл. 1.21**.

Максимальная величина порогового напряжения для вывода $\overline{\text{RESET}}$ V_{RST} составляет $0.85V_{CC}$. Типовое значение пороговой величины напряжения питания V_{POT} составляет 1.4 В, если схема BOD выключена, и 2.2 В, если включена.

Формат регистра состояния MCUSR микроконтроллера ATtiny15L такой же, что и моделей ATtiny12x (**Рис. 1.39**).

ATtiny28x

Структурная схема подсистемы сброса микроконтроллеров моделей ATtiny28x приведена на **Рис. 1.39**.

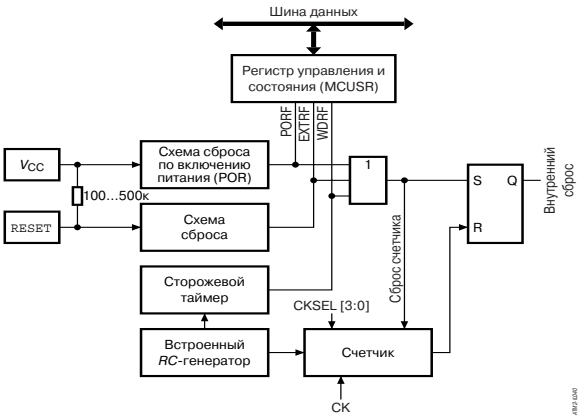


Рис. 1.39. Структурная схема подсистемы сброса микроконтроллеров ATtiny28x

В этих моделях длительность задержки сброса t_{TOUT} определяется только состоянием конфигурационных ячеек CKSEL3...CKSEL0. Соответствующие значения задержек для $V_{CC} = 2.7$ В приведены в **Табл. 1.22**.

Таблица 1.22. **Задержка сброса t_{TOUT} в микроконтроллерах ATtiny28x**

Режим работы тактового генератора	CKSEL 3...0	t_{TOUT} ($V_{CC} = 2.7$ В)
Внешний кварц или керамический резонатор	1111	1K CK*
	1110	4.2 мс + 1K CK
	1101	67 мс + 1K CK
	1100	16K CK
	1011	4.2 мс + 16K CK
	1010	67 мс + 16K CK
Внешний низкочастотный резонатор	1001	67 мс + 1K CK
	1000	67 мс + 32K CK
Внешняя *RC*-цепочка	0111	6 CK
	0110	4.2 мс + 6 CK
	0101	67 мс + 6 CK
Внутренняя *RC*-цепочка	0100	6 CK
	0011	4.2 мс + 6 CK
	0010**	67 мс + 6 CK
Внешний сигнал синхронизации	0001	6 CK
	0000	4.2 мс + 6 CK

* CK — период тактового сигнала микроконтроллера.

** Режим по умолчанию.

Как и в большинстве моделей, для формирования первой части задержки используется *RC*-генератор сторожевого таймера (**Рис. 1.39**). Задержке в 4.2 мс соответствуют 1024 (1K) тактов сигнала генератора, а 67 мс — 16384 (16K) тактов. Поскольку частота *RC*-генератора зависит от напряжения питания микроконтроллера, величина задержки тоже будет зависеть от напряжения питания.

Максимальная величина порогового напряжения для вывода $\overline{\text{RESET}}$ V_{RST} составляет $0.6V_{CC}$. Типовое значение пороговой величины напряжения питания V_{POT} составляет 1.4 В.

В микроконтроллерах ATtiny28x для определения источника сброса используется регистр управления и состояния микроконтроллера MCUCS, расположенный по адресу $07. Формат этого регистра был показан на **Рис. 1.15**, а назначение его разрядов приведено в **Табл. 1.9**.

3.5. Прерывания

Прерывание прекращает нормальный ход программы для выполнения приоритетной задачи, определяемой внутренним или внешним событием микроконтроллера. При возникновении прерывания микроконтроллер сохраняет в стеке содержимое счетчика команд PC и загружает в него адрес соответствующего вектора прерывания. По этому адресу должна находиться команда относительного перехода к подпрограмме обработки прерывания. Кроме того, последней командой подпрограммы обработки прерывания должна быть команда RETI, которая обеспечивает возврат в основную программу и восстановление предварительно сохраненного счетчика команд.

Поскольку источниками прерываний являются различные периферийные устройства микроконтроллеров, количество прерываний (4...8) зависит от конкретной модели.

3.5.1. Таблица векторов прерываний

Микроконтроллеры семейства Tiny имеют многоуровневую систему приоритетных прерываний. Младшие адреса памяти программ, начиная с адреса $001, отведены под таблицу векторов прерывания. Каждому прерыванию соответствует свой адрес в этой таблице, и именно этот адрес загружается в счетчик команд при возникновении прерывания. Положение вектора в таблице также определяет и приоритет соответствующего прерывания: чем меньше адрес, тем выше приоритет прерывания. Размер таблицы зависит от модели микроконтроллера и составляет от 4-х (адреса $001...$004) до 8-ми (адреса $001...$008) векторов. Распределение таблицы векторов прерываний для всех микроконтроллеров семейства приведено в **Табл. 1.23**.

Если в программе, загруженной в микроконтроллер, прерывания никогда не используются, то на месте таблицы векторов прерываний может быть размещена часть этой программы.

Таблица 1.23. **Таблица векторов прерываний**

Источник	Описание	ATtiny							
		11x		12x		15L		28x	
		№	Адрес	№	Адрес	№	Адрес	№	Адрес
INT0	Внешнее прерывание 0	1	$001	1	$001	1	$001	1	$001
INT1	Внешнее прерывание 1	—	—	—	—	—	—	2	$002
PIN_CHANGE	Изменение сигналов на выводах	2	$002	2	$002	2	$002		
LOW_LEVEL	НИЗКИЙ уровень на входе порта B	—	—	—	—	—	—	3	$003
TIMER1 COMPA	Совпадение «A» таймера/счетчика T1	—	—	—	—	3	$003	—	—
TIMER1 OVF	Переполнение таймера/счетчика T1	—	—	—	—	4	$004	—	—
TIMER0 OVF	Переполнение таймера/счетчика T0	3	$003	3	$003	5	$005	4	$004
EE_RDY	EEPROM готово	—	—	4	$004	6	$006	—	—
ANA_COMP	Аналоговый компаратор	4	$004	5	$005	7	$007	5	$005
ADC	Преобразование АЦП завершено	—	—	—	—	8	$008	—	—

3.5.2. Обработка прерываний

Для разрешения прерываний флаг I регистра SREG должен быть установлен в «1». Разрешение или запрещение (маскирование) отдельных прерываний производится установкой или сбросом соответствующих разрядов регистров масок прерываний, рассматриваемых ниже.

Обработка прерываний осуществляется следующим образом:

- при выполнении условий, необходимых для генерации прерывания, соответствующий этому прерыванию флаг устанавливается в «1», а флаг I аппаратно сбрасывается, запрещая тем самым обработку следующих прерываний. Однако в подпрограмме обработки прерывания этот флаг можно будет установить в «1» для разрешения вложенных прерываний;
- если прерывание разрешено (флаг разрешения прерывания установлен), в счетчик команд загружается адрес вектора соответствующего прерывания ($002...$008). При этом флаг прерывания аппаратно сбрасывается. Ряд флагов прерываний могут быть также сброшены записью «1» в разряд регистра, соответствующий флагу;

- если же прерывание запрещено (флаг разрешения прерывания сброшен), флаг прерывания остается в состоянии «1» до разрешения прерывания (в этом случае он будет сброшен аппаратно), либо до программного сброса этого флага;
- выполняется подпрограмма обработки прерывания;
- выполняется команда возврата из прерывания RETI, при этом флаг I аппаратно устанавливается в «1», разрешая обработку последующих прерываний;
- центральный процессор автоматически восстанавливает содержимое счетчика команд. Затем основная программа продолжает свое выполнение с того места, где она была прервана.

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

Следует помнить, что для прерываний, вызываемых статическими событиями (например, для прерывания, генерируемого при равенстве содержимого счетного регистра и регистра сравнения таймера), флаг прерывания устанавливается только в момент возникновения события. Если флаг прерывания сброшен, а условия генерации прерывания присутствуют, флаг будет установлен только в момент возникновения следующего события.

С другой стороны, для внешних прерываний, генерируемых по уровню, флаги не предусмотрены, поэтому информация об этих прерываниях будет храниться до тех пор, пока присутствует событие, вызывающее прерывание.

Микроконтроллеры семейств Tiny поддерживают очередь прерываний, которая работает следующим образом: если условия генерации одного или более прерываний возникают в то время, когда флаг общего разрешения прерываний сброшен (все прерывания запрещены), соответствующие флаги устанавливаются в «1» и остаются в этом состоянии до установки флага общего разрешения прерываний. После разрешения прерываний выполняется их обработка в соответствии с приоритетом.

Наименьшее время отклика для любого прерывания составляет 4 машинных цикла. В течение этих циклов происходит сохранение счетчика команд в стеке. В течение следующих двух циклов выполняется команда перехода к подпрограмме обработки прерывания. Если прерывание произойдет во время выполнения команды длящейся несколько циклов, то генерация прерывания произойдет только после

выполнения этой команды. Если же прерывание произойдет во время нахождения микроконтроллера в «спящем» режиме, время отклика увеличивается еще на 4 машинных цикла.

Возврат в основную программу занимает 4 машинных цикла, в течение которых происходит восстановление счетчика команд из стека. После выхода из прерывания процессор всегда выполняет одну команду основной программы, прежде чем обслужить любое отложенное прерывание.

3.5.3. Внешние прерывания. Регистры GIMSK и GIFR

Регистр GIMSK (General Interrupt MaSK Register — общий регистр маски прерываний), расположенный по адресу $3B, предназначен для управления внешними прерываниями в моделях ATtiny11x, ATtiny12x и ATtiny15L. Формат этого регистра показан на **Рис. 1.40**, а описание его разрядов приведено в **Табл. 1.24**.

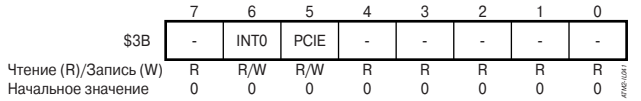


Рис. 1.40. Формат регистра маски прерываний GIMSK

Таблица 1.24. **Разряды регистра маски прерываний GIMSK**

Разряд	Название	Описание
7	—	Не используется, читается как лог. 0
6	INT0	**Разрешение внешнего прерывания INT0.** Если в этом разряде записана лог. 1 и флаг I регистра SREG также установлен в «1», то разрешается внешнее прерывание с вывода INT0. Условие генерации прерывания определяются содержимым разрядов ISC01 и ISC00 регистра MCUCR (см. подраздел 2.2.2.2)
5	PCIE	**Разрешение прерывания по изменению состояния выводов.** Если в этом разряде записана лог. 1 и флаг I регистра SREG также установлен в «1», то разрешается внешнее прерывание по изменению состояния выводов микроконтроллера. К возникновению прерывания приводит любое изменение сигнала на любом входе или контакте ввода/вывода (см. ниже)
4...0	—	Не используются, читаются как лог. 0

Соответственно, регистр GIFR (General Interrupt Flag Register — общий регистр флагов прерываний), расположенный по адресу $3A, предназначен для индикации наступления внешних прерываний. Формат этого регистра показан на **Рис. 1.41**, а описание его разрядов приведено в **Табл. 1.25**.

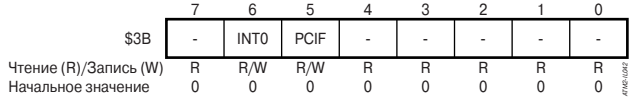


Рис. 1.41. Формат регистра флагов прерываний GIFR

Таблица 1.25. **Разряды регистра флагов прерываний GIFR**

Разряд	Название	Описание
7	—	Не используется, читается как лог. 0
6	INTF0	**Флаг внешнего прерывания INT0.** Если в результате события на выводе INT0 сформировался запрос на внешнее прерывание, этот разряд устанавливается в 1. Флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания или программно, записью в него лог. 1. Флаг INTF0 сброшен постоянно, если генерация прерывания должна происходить по НИЗКОМУ уровню на выводе INT0
5	PCIF	**Флаг прерывания по изменению состояния выводов.** Если в результате события на любом выводе сформировался запрос на прерывание, этот разряд устанавливается в «1». Флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания или программно, записью в него лог. 1
4...0	—	Не используются, читаются как лог. 0

Следует иметь в виду, что для программного сброса одного флага нельзя использовать команду SBI (установить разряд регистра ввода/вывода), т. к. при ее выполнении будут сброшены оба флага. Это связано с тем, что команда сначала считывает содержимое регистра, затем изменяет указанный разряд и записывает полученный результат обратно в регистр. Аналогично при выполнении команды CBI (сбросить разряд регистра ввода/вывода), будут сброшены все разряды регистра, кроме указанного.

Следует отметить, что все внешние прерывания генерируются даже в том случае, если соответствующие выводы сконфигурированы как выходы. Эта особенность микроконтроллеров позволяет генерировать прерывания программно.

Приведем несколько замечаний о прерывании, возникающем при изменении состояния выводов микроконтроллера:

- для выводов PB2...PB0 (модели ATtiny11x/ATtiny12x) и выводов PB4...PB0 (модель ATtiny15L) прерывание возникает всегда при изменении состояния любого из указанных выводов;
- для выводов PB5...PB3 (модели ATtiny11x/ATtiny12x) и вывода PB5 (модель ATtiny15L) прерывание возникает при изменении состояния любого из указанных выводов только в том случае, если он функционирует как контакт ввода/вывода;
- прерывание возникает, даже если изменение сигнала на выводе приводит к генерации другого прерывания, например, прерывания INT0. То есть изменение сигнала на выводе INT0 может привести к одновременному возникновению двух прерываний;
- генерация прерывания гарантируется только для импульсов, длительность которых превышает 1 период тактового сигнала микроконтроллера.

3.5.4. Прерывания от таймеров. Регистры TIMSK и TIFR

Регистр TIMSK (Timer/Counter Interrupt MaSK — регистр маски прерываний от таймеров/счетчиков), расположенный по адресу $39, предназначен для управления прерываниями от таймеров в моделях ATtiny11x, ATtiny12x и ATtiny15L. Формат этого регистра для различных моделей показан на **Рис. 1.42**, а описание его разрядов приведено в **Табл. 1.26**.

	7	6	5	4	3	2	1	0	
$39	-	-	-	-	-	-	TOIE0	-	ATtiny11x ATtiny12x
Чтение (R)/Запись (W)	R	R	R	R	R	R	R/W	R	
Начальное значение	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
$39	-	OCIE1A	-	-	-	TOIE1	TOIE0	-	ATtiny15L
Чтение (R)/Запись (W)	R	R/W	R	R	R	R/W	R/W	R	
Начальное значение	0	0	0	0	0	0	0	0	

Рис. 1.42. Формат регистра маски прерываний от таймеров/счетчиков TIMSK

Таблица 1.26. **Разряды регистра маски прерываний от таймеров/счетчиков TIMSK**

Разряд	Название	Описание	Модель
7	—	Не используется, читается как «0»	Все модели
6	OCIE1A	Флаг разрешения прерывания по событию «Compare Match A» («совпадение A») таймера/счетчика T1	ATtiny15L
	—	Не используется, читается как «0»	ATtiny11x ATtiny12x
5..3	—	Не используется, читается как «0»	Все модели
2	TOIE1	Флаг разрешения прерывания по переполнению таймера/счетчика T1	ATtiny15L
	—	Не используется, читается как «0»	ATtiny11x ATtiny12x
1	TOIE0	Флаг разрешения прерывания по переполнению таймера/счетчика T0	Все модели
0	—	Не используется, читается как «0»	Все модели

Регистр TIFR (Timer/Counteer Interrupt Flag Register — регистр флагов прерываний от таймеров/счетчиков), расположенный по адресу $38, предназначен для индикации наступления прерываний от таймеров. Формат этого регистра для различных моделей показан на **Рис. 1.43**, а описание его разрядов приведено в **Табл. 1.27**. Каждый флаг этого регистра сбрасывается аппаратно при запуске подпрограммы обработки соответствующего прерывания или программно, записью в него лог. 1. Другие особенности функционирования TIFR аналогичны особенностям функционирования GIFR.

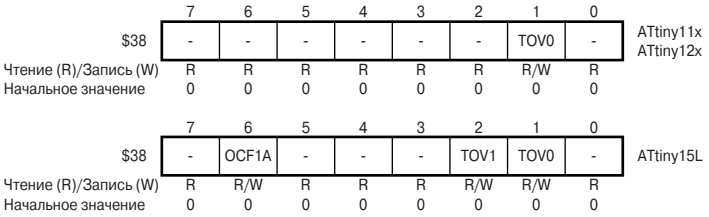


Рис. 1.43. Формат регистра флагов прерываний от таймеров/счетчиков TIFR

Таблица **Разряды регистра флагов прерываний от таймеров/счетчиков TIFR**

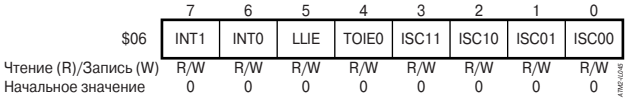
Разряд	Название	Описание	Модель
7	—	Не используется, читается как «0»	Все модели
6	OCF1A	Флаг устанавливается в «1» при совпадении значения таймера/счетчика T1 с содержимым регистра OCR1A (событие «совпадение A»)	ATtiny15L
	—	Не используется, читается как «0»	ATtiny11x ATtiny12x
5..3	—	Не используется, читается как «0»	Все модели
2	TOV1	Флаг устанавливается в «1» при переполнении таймера/счетчика T1	ATtiny15L
	—	Не используется, читается как «0»	ATtiny11x ATtiny12x
1	TOV0	Флаг устанавливается в «1» при переполнении таймера /счетчика T0	Все модели
0	—	Не используется, читается как «0»	Все модели

Более подробная информация о функционировании таймеров и о событиях, вызываемых ими, приведена в Главе 5.

3.5.5. Управление прерываниями в микроконтроллерах ATtiny28x. Регистры ICR и IFR

В микроконтроллерах ATtiny28x для управления внешними прерываниями и прерываниями от таймеров используются одни и те же регистры ввода/вывода. Всего этих регистров два — регистр маски и регистр флагов.

Регистр ICR (Interrupt Control Register — регистр управления прерываниями), расположенный по адресу $06, предназначен для разрешения/запрещения отдельных прерываний в моделях ATtiny28x. Формат этого регистра показан на **Рис. 1.44**, а описание его разрядов приведено в **Табл. 1.28**.

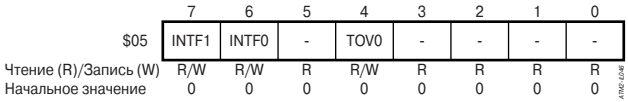


Формат регистра управления прерываниями ICR

Таблица **Разряды регистра управления прерываниями ICR**

Разряд	Название	Описание		
7	INT1	**Разрешение внешнего прерывания INT1.** Если в этом разряде записана лог. 1 и флаг I регистра SREG также установлен в «1», то разрешаются внешние прерывания с вывода INT1. Условия генерации прерывания определяются содержимым разрядов ISC11 и ISC10 регистра		
6	INT0	**Разрешение внешнего прерывания INT0.** Если в этом разряде записана лог. 1 и флаг I регистра SREG также установлен в «1», то разрешаются внешние прерывания с вывода INT0. Условия генерации прерывания определяются содержимым разрядов ISC01 и ISC00 регистра		
5	LLIE	**Разрешение прерывания по НИЗКОМУ уровню на входе.** Если в этом разряде записана лог. 1 и флаг I регистра SREG также установлен в «1», то разрешается внешнее прерывание по НИЗКОМУ уровню на входе микроконтроллера. Прерывание происходит при подаче на любой вывод порта B сигнала НИЗКОГО уровня при условии, что этот вывод не используется для каких-либо специальных целей. Например, если аналоговый компаратор включен, появление НИЗКОГО уровня на выводе PB0 или PB1 не приведет к генерации прерывания. То же касается и спец. функций T0, INT0 и INT1		
4	TOIE0	**Разрешение прерывания по переполнению таймера/счетчика T0.** Если в этом разряде записана лог. 1 и флаг I регистра SREG также установлен в «1» разрешается прерывание по переполнению таймера /счетчика T0		
3, 2	ISC11, ISC10	**Условие генерации внешнего прерывания INT1**		
		ISC11	ISC10	Условие
		0	0	По НИЗКОМУ уровню на выводе INT1
		0	1	При любом изменении сигнала на выводе INT1
		1	0	По спадающему фронту сигнала на выводе INT1
		1	1	По нарастающему фронту сигнала на выводе INT1
1, 0	ISC01, ISC00	**Условие генерации внешнего прерывания INT0**		
		ISC01	ISC00	Условие
		0	0	По НИЗКОМУ уровню на выводе INT0
		0	1	При любом изменении сигнала на выводе INT0
		1	0	По спадающему фронту сигнала на выводе INT0
		1	1	По нарастающему фронту сигнала на выводе INT0

Соответственно регистр IFR (Interrupt Flag Register — регистр флагов прерываний), расположенный по адресу $05, предназначен для индикации наступления отдельных прерываний. Формат этого регистра показан на **Рис. 1.45**, а описание его разрядов приведено в **Табл. 1.29**. Другие особенности функционирования регистра IFR аналогичны особенностям функционирования регистра GIFR.



Формат регистра флагов прерываний IFR

Таблица **Разряды регистра флагов прерываний IFR**

Разряд	Назвнание	Описание
7	INTF1	**Флаг внешнего прерывания INT1.** Если в результате события на выводе INT1 сформировался запрос на внешнее прерывание, этот разряд устанавливается в «1». Флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания или программно, записью в него лог. 1. Флаг INTF1 сброшен постоянно, если генерация прерывания должна происходить по НИЗКОМУ уровню на выводе INT1
6	INTF0	**Флаг внешнего прерывания INT0.** Если в результате события на выводе INT0 сформировался запрос на внешнее прерывание, этот разряд устанавливается в «1». Флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания или программно, записью в него лог. 1. Флаг INTF0 сброшен постоянно, если генерация прерывания должна происходить по НИЗКОМУ уровню на выводе INT0
5	—	Не используется, читается как «0»
4	TOV0	**Флаг прерывания по переполнению таймера/счетчика T0.** Флаг устанавливается в «1» при переполнении таймера/счетчика T0
3...0	—	Не используются, читаются как «0»

Как и в остальных моделях, внешние прерывания INT0 и INT1 генерируются даже в том случае, если соответствующие выводы сконфигурированы как выходы. Эта особенность микроконтроллеров позволяет генерировать прерывания программно.

Глава 4. **Порты ввода/вывода**

4.1. Общие сведения

Как и любые другие микроконтроллеры, микроконтроллеры семейства Tiny имеют порты ввода/вывода. Каждый порт состоит из определенного числа выводов, через которые микроконтроллер может принимать или передавать цифровые сигналы. Задание направления передачи данных через любой контакт ввода/вывода может быть произведено программно в любой момент времени. При этом некоторые модели имеют выводы, способные работать либо только как входы, либо только как выходы. Входные буферы портов построены по схеме триггера Шмитта. Для большинства линий, сконфигурированных как входные, также имеется возможность подключения внутреннего подтягивающего резистора сопротивлением 35...120 кОм между входом и шиной питания $_{CC}$. Кроме того, если между входом с задействованным внутренним подтягивающим резистором и общей шиной подключить нагрузку, этот вход может служить источником тока.

В портах ввода/вывода микроконтроллеров семейства Tiny реализована истинная функциональность вида «чтение/модификация/запись». Благодаря этому, используя команды SBI и CBI, можно выполнять операции над любым выводом, не воздействуя на другие выводы порта. Это относится к изменению режима работы контакта ввода/вывода, к изменению выходного значения и к изменению состояния внутреннего подтягивающего резистора (для входов).

Отличительной особенностью моделей семейства, выпускающихся в 8-выводном корпусе, является совмещение функций ввода/вывода с функциями управления (выводы для подключения внешнего резонатора и вывод сброса). Соответственно в распоряжении пользователя оказываются до 6 линий ввода/вывода, т. к. эти модели могут быть сконфигурированы для работы без внешнего резонатора и без вывода аппаратного сброса.

Микроконтроллеры из каждой группы моделей семейства имеют различное количество портов и соответственно контактов ввода/вывода:

- ATtiny11x имеют один 6-разрядный порт ввода/вывода (порт B). Контактов ввода/вывода — 5, входных контактов — 1;
- ATtiny12x имеют один 6-разрядный порт ввода/вывода (порт B). Контактов ввода/вывода — 6;
- ATtiny15L также имеет один 6-разрядный порт ввода/вывода B. Контактов ввода/вывода — 6;
- ATtiny28x имеют три порта ввода/вывода: порт A (4-разрядный), порт B (8-разрядный) и порт D (8-разрядный). Общее количество контактов ввода/вывода равно 11, выходных контактов — 1 (линия порта A), входных контактов — 8 (порт B).

Во всех микроконтроллерах семейства подавляющее большинство контактов ввода/вывода имеют дополнительные функции и используются периферийными устройствами микроконтроллеров.

4.2. Обращение к портам ввода/вывода

Обращение к портам производится через регистры ввода/вывода. Под каждый порт (за исключением порта B в модели ATtiny28x) в адресном пространстве ввода/вывода зарезервировано по 3 адреса. По этим адресам размещаются три регистра: регистр данных порта, регистр направления данных (для порта A в моделях ATtiny28x — регистр управления) и регистр выводов порта. Поскольку все выводы порта B моделей ATtiny28x могут работать только как входы, этому порту сопоставлен только один регистр — регистр выводов порта.

Адреса всех регистров, относящихся к портам ввода/вывода, приведены в **Табл. 1.30**.

Таблица. 1.30. **Регистры портов ввода/вывода**

Порт	Название	Функция	Адрес	ATtiny			
				11x	12x	15L	28L
A	PORTA	Регистр данных порта A	$1B	—	—	—	♦
	PACR	Регистр управления порта A	$1A	—	—	—	♦
	PINA	Регистр выводов порта A	$19	—	—	—	♦
B	PORTB	Регистр данных порта B	$18	♦	♦	♦	—
	DDRB	Регистр направления порта B	$17	♦	♦	♦	—
	PINB	Регистр выводов порта B	$16	♦	♦	♦	♦
D	PORTD	Регистр данных порта D	$12	—	—	—	♦
	DDRD	Регистр направления порта D	$11	—	—	—	♦
	PIND	Регистр выводов порта D	$10	—	—	—	♦

Вообще говоря, «регистры» PIN*x* регистрами не являются, по этим адресам осуществляется доступ к физическим значениям сигналов на выводах порта. Соответственно они доступны только для чтения, тогда как остальные регистры доступны и для чтения, и для записи.

Таким образом, запись в порт означает запись требуемого состояния для каждого вывода порта в соответствующий регистр данных порта PORT*x*. А чтение состояния порта выполняется чтением либо регистра данных порта PORT*x*, либо регистра выводов порта PIN*x*. При чтении регистра выводов порта PIN*x* происходит считывание сигналов, присутствующих на выводах порта. А при чтении регистра данных порта PORT*x* происходит считывание данных, находящихся в регистре-защелке порта (как для входных, так и для выходных контактов). При нахождении микроконтроллера в состоянии сброса (см. Главу 3) выводы всех портов находятся в третьем состоянии (состояние высокого импеданса Hi-Z).

4.3. Конфигурирование портов ввода/вывода

Порты ввода/вывода (и даже отдельные разряды одноименных портов) разных моделей семейства имеют различные возможности по конфигурированию. Рассмотрение этих возможностей начнем с младших моделей семейства Tiny (ATtiny11x, ATtiny12x и ATtiny15L), имеющих в своем составе единственный 6-разрядный порт ввода/вывода (порт B). Формат всех регистров, относящихся к этому порту, приведен на **Рис. 1.46**.

	7	6	5	4	3	2	1	0	
$18	-	-	-	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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	
	7	6	5	4	3	2	1	0	
$17	-	-	(DDB5)	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Чтение (R)/Запись (W)	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	
	7	6	5	4	3	2	1	0	
$16	-	-	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Чтение (R)/Запись (W)	R	R	R	R	R	R	R	R	
Начальное значение	0	0	N/A	N/A	N/A	N/A	N/A	N/A	

Замечание: в моделях ATiny11x разряд DDB5 регистра DDRB отсутствует

Рис. 1.46. Формат регистров PORTB, DDRB и PINB в моделях ATtiny11x/12x/15L

Направление передачи данных определяется содержимым регистра направления данных DDRB. Если разряд DDB*n* этого регистра установлен в «1», соответствующий *n*-й вывод порта является выходом. Если же разряд DDB*n* этого регистра сброшен в «0», соответствующий вывод порта является входом.

В моделях ATtiny12x и ATtiny15L используются 5 младших разрядов этого регистра, а в моделях ATtiny11x только 4 (вывод PB5 может работать только как вход). В моделях ATtiny12x и ATtiny15L вывод PB5 может функционировать либо как вход, либо как выход с открытым стоком (на выходе PB5 может присутствовать только сигнал лог. 0).

Максимальная нагрузочная способность выходов PB4...PB0 составляет 20 мА, а вывода PB5 (ATtiny12x и ATtiny15L) — 12 мА.

Управление внутренними подтягивающими резисторами в моделях ATtiny12x и ATtiny15L осуществляется на двух уровнях. Общее управление осуществляется с помощью разряда PUD регистра MCUCR. Если этот разряд установлен в «1», использование подтягивающих резисторов запрещено (они отключены от всех выводов порта). Если же этот разряд сброшен в «0», использование подтягивающих резисторов разрешено. Управление подтягивающими резисторами в этом случае осуществляется с помощью регистра данных PORTB индивидуально для каждого из контактов PB4...PB0 порта (вывод PB5 не имеет внутреннего подтягивающего резистора). Соответственно используются только 4 младших разряда регистра.

Внутренний подтягивающий резистор подключается между выводом и шиной питания, если этот вывод является входом и соответствующий ему разряд PORTB*n* регистра PORTB установлен в «1». Отключить подтягивающий резистор можно сбросом разряда регистра PORTB, переключением вывода в режим выхода либо записью лог. 1 в разряд PUD регистра MCUCR.

В моделях ATtiny11x управление внутренними подтягивающими резисторами осуществляется только с помощью регистра данных PORTB, как описано выше.

Все возможные конфигурации выводов порта B приведены в **Табл. 1.31** (*x* — B).

Таблица 1.31. **Зависимость конфигурации выводов порта от состояния разрядов регистров DDR*x* и PORT*x***

DDx*n**	PORT *xn*	Функция вывода	Резистор	Примечания
0	0	Вход	Отключен	Третье состояние (Hi-Z)**
0	1	Вход	Подключен	При подключении нагрузки между выводом и общим проводом, вывод является источником тока. В моделях ATtiny12x и ATtiny15L резистор может быть отключен записью лог. 1 в разряд PUD регистра MCUCR
1	0	Выход	Отключен	Выход установлен в «0»
1	1	Выход	Отключен	Выход установлен в «1»

* *n* – номер вывода (разряд порта).

** Состояние выводов порта при сбросе.

Следует помнить, что если вывод используется каким-либо периферийным устройством микроконтроллера, то соответствующие этому выводу разряды регистров DDRB и PORTB должны быть установлены в соответствии с дополнительной функцией вывода. В то же время для выводов, используемых как вывод сброса или для подключения внешнего резонатора, содержимое соответствующих разрядов этих регистров игнорируется.

Теперь рассмотрим конфигурирование портов ввода/вывода в микроконтроллерах ATtiny28x.

Порт A

Это 4-разрядный порт с двунаправленными выводами PA3, PA1 и PA0 и однонаправленным (только выход) выводом PA2. Управление выводом осуществляется с помощью регистра управления PACR, формат которого приведен на **Рис. 1.47**.

	7	6	5	4	3	2	1	0
$1A	-	-	-	-	DDA3	PA2HC	DDA1	DDA0
Чтение (R)/Запись (W)	R	R	R	R	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Рис. 1.47. Формат регистра PACR моделей ATtiny28x

Состояние разрядов DDA3, DDA1 и DDA0 определяет направление передачи данных через соответствующий вывод порта. Если разряд установлен в «1», вывод работает как выход. Если же разряд сброшен в «0», вывод работает как вход. 2-й разряд регистра (PA2HC) управляет нагрузочной способностью выхода PA2, который предназначен специально для управления светодиодным индикатором. Если разряд сброшен в «0», максимальная нагрузочная способность вывода составляет 15 мА (при V_{CC} = 1.8 В). При записи в этот разряд «1» к выходу подключается дополнительный драйвер, и максимальная нагрузочная способность увеличивается до 25 мА при V_{CC} = 1.8 В.

Управление внутренними подтягивающими резисторами для выводов PA3, PA1 и PA0 осуществляется с помощью регистра данных PORTA. Если вывод является входом, то при установке в «1» соответствующего ему разряда PORTA*n* между выводом и шиной питания подключается внутренний подтягивающий резистор. При сбросе этого разряда в «0» или при переключении вывода в режим выхода подтягивающий резистор отключается. Возможные конфигурации выводов порта такие же, как и в младших моделях семейства (см. **Табл. 1.31**, *x* — A).

Вывод PA2 микроконтроллера может управляться не только программно (посредством регистра PORTA), но и аппаратно. Для аппаратного управления состоянием этого вывода в составе микроконтроллеров ATtiny28x имеется специальное периферийное устройство, называемое аппаратным модулятором (Hardware Modulator). Подробно об этом устройстве будет рассказано в следующем параграфе.

Порт B

Все выводы порта B могут работать только как входы, соответственно пользователь может управлять только подключением или отключением внутренних подтягивающих резисторов (одновременно для всех выводов порта). Для этой цели служит 7-й разряд (PLUPB) регистра MCUSR (см. подраздел 2.2.2.2).

Если этот разряд установлен в «1», внутренние подтягивающие резисторы на всех входах порта B подключены, если сброшен — отключены. При использовании какого-либо вывода порта периферийным устройством соответствующий подтягивающий резистор отключается автоматически независимо от состояния разряда PLUPB.

Порт D

Порт D является обыкновенным 8-разрядным портом ввода/вывода с подключаемыми внутренними подтягивающими резисторами. Управление выводами порта осуществляется также, как и управление выводами PA4, PA2 и PA1 порта A. Направление передачи данных определяется содержимым регистра направления данных DDRD, а управление внутренними подтягивающими резисторами осуществляется индивидуально для каждого вывода с помощью регистра данных PORTD. В регистрах PORTD, PIND и DDRD задействованы все разряды, поскольку порт D 8-разрядный. Все возможные конфигурации выводов порта D приведены в **Табл. 1.31** (x — D).

Максимальная нагрузочная способность выходов порта D составляет 10 мА.

4.4. Аппаратный модулятор

Аппаратный модулятор (Hardware Modulator) предназначен для управления выводом PA2 микроконтроллеров ATtiny28x. Единственной задачей модулятора является генерирование последовательности импульсов с заданными параметрами.

Структурная схема блока аппаратного модулятора приведена на **Рис. 1.48**.

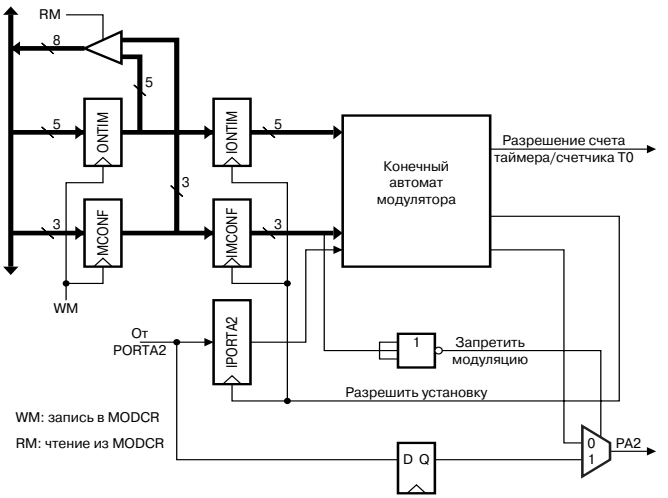


Рис. 1.48. Структурная схема аппаратного модулятора

Для управления параметрами генерируемого сигнала предназначен регистр MODCR (MODulation Control Register), расположенный по адресу $02. Формат этого регистра приведен на **Рис. 1.49**.

	7	6	5	4	3	2	1	0
$02	ONTIM4	ONTIM3	ONTIM2	ONTIM1	ONTIM0	MCONF2	MCONF1	MCONF0
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. 1.49. Формат регистра MODCR

Старшие 5 разрядов регистра (ONTIM4...ONTIM0) определяют время нахождения выхода PA2 в активном (НИЗКИЙ уровень) состоянии. Число машинных циклов микроконтроллера, в течение которых вывод PA2 будет находиться в активном состоянии, на единицу больше значения, записанного в разрядах ONTIM4...ONTIM0.

Младшие три разряда регистра (MCONF2...MCONF0) определяют соотношение между длительностями активного и неактивного состояния вывода PA2 и соответственно скважность генерируемого сигнала. Значения скважности сигнала для различного содержимого этих разрядов, а также возможные значения периода генерируемого сигнала приведены в **Табл. 1.32**.

Таблица 1.32. **Зависимость скважности и периода модуляции от содержимого разрядов MCONF2...MCONF0**

MCONF 2...0	Длительность сигнала		Скважность	Период модуляции*		Пояснения
	«0»	«1»		min	max	
000	X	X	1	X	X	Нет модуляции
001	ONTIM + 1	ONTIM + 1	0.5	2 CK	64 CK	
010	ONTIM + 1	2(ONTIM + 1)	0.33	3 CK	96 CK	
011	ONTIM + 1	3(ONTIM + 1)	0.25	4 CK	128 CK	
100	2(ONTIM + 1)	ONTIM + 1	0.67	3 CK	96 CK	
101	3(ONTIM + 1)	ONTIM + 1	0.75	4 CK	128 CK	
110	Зарезервировано					
111	X	X	**	1 CK	1 CK	Сигнал высокой частоты

* Минимальный период модуляции соответствует ONTIM = 0, а максимальный — ONTIM = 31.

** Зависит от параметров тактового сигнала микроконтроллера, т. к. сигнал на выходе PA2 идентичен ему.

Несущая частота (частота генерируемого сигнала) определяется соотношением $f_C = f_{OSC}/(T_{ON} + T_{OFF})$, где f_{OSC} — тактовая частота микроконтроллера, T_{ON} — время нахождения вывода PA2 в активном состоянии, T_{OFF} — время нахождения вывода PA2 в выключенном состоянии. При значении MCONF = 111 частота сигнала равна частоте тактового сигнала микроконтроллера.

Для примера в **Табл. 1.33** приводятся значения рассмотренных параметров для «типовых» значений частот тактового и генерируемого сигналов (см. также **Рис. 1.50**).

Таблица 1.33. **Типовые конфигурации модулятора**

Частота резонатора [МГц]	Несущая частота [кГц]	Погрешность установки частоты [%]	Скважность	Значения параметров	
				ONTIM	MCONF
0.455	38	0.2	0.25	2	011
0.455	38	0.2	0.33	3	010
0.455	38	0.2	0.50	5	001
0.455	38	0.2	0.67	3	100
0.455	38	0.2	0.75	2	101
1	38	1.2	0.50	12	001
1.8432	38	1.1	0.25	11	011
1.8432	38	1.1	0.33	15	010
1.8432	38	1.1	0.50	23	001
2	38	1.2	0.25	12	011
2	38	1.2	0.50	25	001

Продолжение таблицы 1.33

Частота резонатора [МГц]	Несущая частота [кГц]	Погрешность установки частоты [%]	Скважность	Значения параметров	
				ONTIM	MCONF
2.4576	38	1.1	0.50	31	001
3.2768	38	2.0	0.25	21	011
4	38	1.2	0.25	25	011
0.455	455	0.0	≈ 0.50	X	111
1	455	9.9	0.50	0	001
1.82	455	0.0	0.25	0	011
1.82	455	0.0	0.50	1	001
1.8432	455	1.3	0.25	0	011
1.8432	455	1.3	0.50	1	001
2	455	9.9	0.25	0	011
2	455	9.9	0.50	1	001
2.4576	455	10.0	0.33	1	010
2.4576	455	10.0	0.50	2	001
3.2768	455	10.0	0.25	1	011
3.2768	455	10.0	0.50	3	001
3.64	455	0.0	0.25	1	011
3.64	455	0.0	0.50	3	001
4	455	9.9	0.25	1	011
4	455	9.9	0.50	3	001

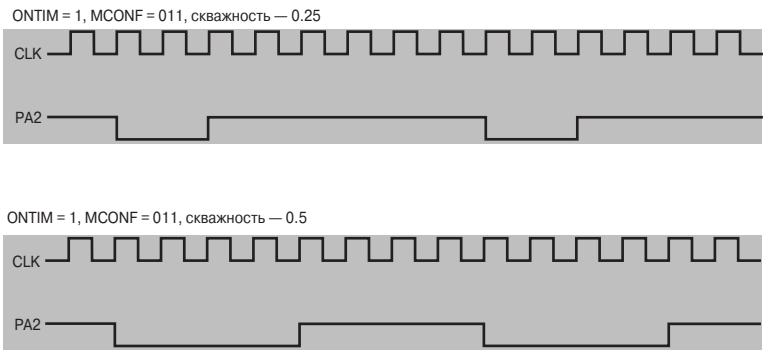


Рис. 1.50. Примеры сигналов, формируемых аппаратным модулятором

Разумеется, состояние выхода PA2 может определяться не только аппаратным модулятором. Все возможные условия функционирования выхода PA2 приведены в **Табл. 1.34**.

Таблица 1.34. **Функционирование выхода PA2**

PORTA2	MCONF2...MCONF0	Состояние выхода PA2
0	000	0
0	001...111	Определяется модулятором
1	X	1

Аппаратный модулятор имеет еще одну полезную функцию. Помимо управления состоянием вывода PA2 микроконтроллера, выходной сигнал аппаратного модулятора может использоваться в качестве входного для предделителя таймера/счетчика T0. В этом случае таймер/счетчик может использоваться как для ограничения длительности посылки, так и для задания интервала между посылками. Последнее возможно благодаря тому, что сигнал с выхода модулятора может подаваться на вход предделителя таймера/счетчика, даже если разряд PORTA2 установлен в «1» и генерация отсутствует.

Для использования таймера/счетчика T0 в описанном режиме он должен быть сконфигурирован таким образом, чтобы по его переполнению изменялось состояние выхода PA2. Подробно о конфигурировании таймера см. в Главе 5.

Перед запуском модулятора следует занести требуемые значения в регистр MODCR. Далее запуск может быть осуществлен двумя способами: сбросом разряда PORTA2 регистра PORTA или заданием конфигурации таймера/счетчика T0 таким образом, чтобы он сбросил выход PA2 при переполнении.

Остановить генерацию можно также двумя способами: вручную или с помощью таймера (установка выхода PA2 при переполнении).

Глава 5. **Таймеры в микроконтроллерах семейства Tiny**

5.1. Общие сведения

Все микроконтроллеры семейства Tiny имеют в своем составе один или два (модель ATtiny15L) 8-разрядных таймера/счетчика общего назначения. Первый таймер (таймер T0), имеющийся во всех моделях, может использоваться либо для отсчета и измерения временных интервалов, либо для счета внешних импульсов. При переполнении счетного регистра таймера генерируется запрос на прерывание. Второй таймер (таймер T1) может генерировать запрос на прерывание не только при переполнении счетного регистра, но и при достижении счетным регистром заданного значения. Кроме того, этот таймер может использоваться для генерации сигнала с ШИМ.

Также в составе всех моделей семейства имеется сторожевой таймер, который является непременным атрибутом всех современных микроконтроллеров и используется для предотвращения зацикливания программы.

5.2. Назначение выводов таймеров/счетчиков

В некоторых режимах оба таймера/счетчика используют выводы микроконтроллера, являющиеся линиями портов ввода/вывода общего назначения. Названия этих выводов, а также их функции при работе совместно с таймерами/счетчиками приведены в **Табл. 1.35**.

Следует помнить, что при использовании контактов ввода/вывода совместно с таймерами/счетчиками необходимо самостоятельно сконфигурировать эти выводы в соответствии с их функциональным назначением (вход/выход).

Таблица 1.35. **Выводы, используемые таймерами/счетчиками общего назначения**

Название	ATtiny11x	ATtiny12x	ATtiny15L	ATtiny28x	Описание
T0	PB2	PB2	PB2	PB2	Вход внешнего сигнала таймера T0
OC1A	—	—	PB1	—	Выход схемы сравнения таймера T1
IR	—	—	—	PA2	Выход переполнения таймера T0

5.3. Таймер/счетчик T0

Таймер/счетчик T0 может использоваться для формирования временных интервалов или для подсчета числа внешних событий. Структурная схема таймера/счетчика T0 приведена на **Рис. 1.51**. В его состав входят 2 регистра (регистр управления TCCR0 и счетный регистр TCNT0), а также блок управления таймером. В модели ATtiny15L для управления таймером/счетчиком также используется регистр специальных функций SFIOR. Флаг переполнения счетного регистра таймера TOV0 находится в регистре TIFR (IFR для ATtiny28x). Разрешение/запрещение прерываний от таймера осуществляется установкой/сбросом флага TOIE0 регистра TIMSK (ICR для ATtiny28x).

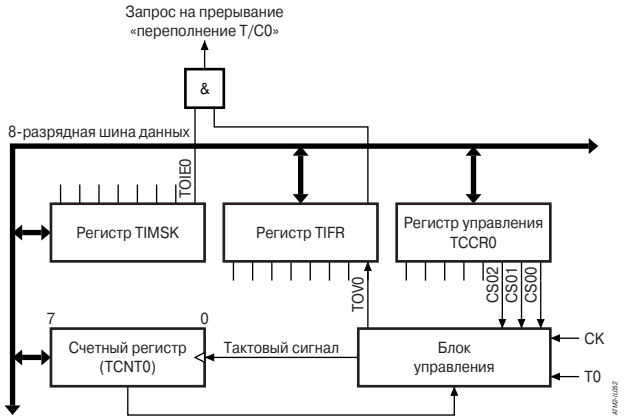


Рис. 1.51. Структурная схема таймера/счетчика T0

Счетный регистр таймера/счетчика TCNT0 расположен по адресу $32 (в моделях ATtiny28x — по адресу $03) и доступен в любой момент времени как для чтения, так и для записи. При записи в регистр

TCNT0 во время работы таймера, счет будет продолжен в следующем за командой записи машинном цикле. После подачи напряжения питания в регистре TCNT0 находится нулевое значение.

При переходе таймера/счетчика из состояния «$FF» в состояние «$00» устанавливается флаг TOV0 регистра TIFR (IFR для ATtiny28x) и генерируется запрос на прерывание. Разрешение прерывания осуществляется установкой в «1» разряда TOIE0 регистра TIMSK (ICR для ATtiny28x) при условии, что флаг общего разрешения прерываний I регистра SREG также установлен в «1».

Таймер/счетчик T0 может работать в режиме таймера или в режиме счетчика событий. В режиме таймера на вход таймера/счетчика поступают импульсы тактового сигнала микроконтроллера (непосредственно или через предделитель). В микроконтроллерах ATtiny28x, кроме того, входным сигналом таймера/счетчика может являться сигнал от аппаратного модулятора. В режиме счетчика событий инкремент содержимого счетного регистра производится по активному фронту сигнала на входе T0 микроконтроллера.

Управление таймером/счетчиком T0 осуществляется с помощью регистра управления таймером TCCR0, расположенного по адресу $33 (в моделях ATtiny28x — по адресу $04). Формат этого регистра для различных моделей семейства приведен на **Рис. 1.52**. Вкратце назначение разрядов этого регистра описано в **Табл. 1.36**.

	7	6	5	4	3	2	1	0	
$33	-	-	-	-	-	CS02	CS01	CS00	ATtiny11x ATtiny12x ATtiny15L
Чтение (R)/Запись (W)	R	R	R	R	R	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
$04	FOV0	-	-	OOM01	OOM00	CS02	CS01	CS00	ATtiny28x
Чтение (R)/Запись (W)	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Рис. 1.52. Формат регистра TCCR0

Выбор режима работы (источника тактового сигнала), а также запуск и остановка таймера/счетчика осуществляются с помощью разрядов CS02...CS00 регистра TCCR0. Соответствие между состоянием этих разрядов и режимом работы таймера/счетчика приведено в **Табл. 1.37**.

Таблица 1.36. **Разряды регистра TCCR0**

Разряд	Название	Описание	Модель
7	FOV0	Принудительное формирование переполнения (1 – сформировать переполнение)	ATtiny28x
6, 5	—	Зарезервированы	Все
4, 3	OOM01, OOM00	Задание режима работы вывода PA2	ATtiny28x
	—	Зарезервированы	ATtiny11x/12x/15L
2...0	CS02...CS00	Выбор источника тактового сигнала	Все

Таблица 1.37. **Выбор источника тактового сигнала для таймера/счетчика T0**

CS02	CS01	CS00	Источник тактового сигнала
0	0	0	Таймер/счетчик остановлен
0	0	1	CK (тактовый сигнал микроконтроллера)
0	1	0	CK/8 (ATtiny11x/12x, ATtiny15L)
			Выход аппаратного модулятора (ATtiny28x)
0	1	1	CK/64
1	0	0	CK/256
1	0	1	CK/1024
1	1	0	Вывод T0, инкремент счетчика производится по спадающему фронту импульсов
1	1	1	Вывод T0, инкремент счетчика производится по нарастающему фронту импульсов

Микроконтроллер ATtiny15L позволяет управлять не только коэффициентом деления предделителя, но и сбрасывать его. Сброс осуществляется записью лог. 1 в разряд PSR0 регистра специальных функций SFIOR (см. ниже).

Необходимо понимать, что сигнал, присутствующий на выводе T0 (при использовании таймера/счетчика в режиме счета внешних событий), синхронизируется с частотой тактового генератора микроконтроллера (состояние вывода T0 считывается по нарастающему фронту

внутреннего тактового сигнала). Поэтому для обеспечения корректной работы таймера от внешнего сигнала промежуток времени между соседними импульсами должен быть больше периода тактового сигнала микроконтроллера.

Следует отметить также, что инкремент содержимого счетного регистра таймера/счетчика при работе в режиме счета внешних событий производится даже в том случае, если вывод T0 сконфигурирован как выход. Эта особенность дает пользователю возможность программно управлять процессом счета.

В микроконтроллерах ATtiny28x таймер/счетчик T0 может также управлять и состоянием вывода PA2. Для управления этой функцией в регистре TCCR0 моделей ATtiny28x задействовано 3 дополнительных разряда (см. **Рис. 1.51**).

Разряд FOV0 предназначен для реализации принудительного переполнения (Force Overflow). При записи в этот разряд лог. 1 вывод PA2 устанавливается в состояние, определяемое содержимым разрядов OOM01 и OOM00 (см. ниже), однако прерывание не генерируется. При чтении этого разряда всегда возвращается «0».

Разряды OOM01 и OOM00 определяют состояние, в которое устанавливается выход PA2 при переполнении (обычном или принудительном) таймера/счетчика T0. Соответствие между содержимым этих разрядов и поведением выхода PA2 приведено в **Табл. 1.38**.

Таблица 1.38. **Выбор режима работы выхода переполнения**

OOM01	OOM00	Действие
0	0	Таймер/счетчик T0 отключен от вывода PA2
0	1	Сигнал на выходе PA2 изменяется на противоположный
1	0	Выход PA2 устанавливается в состояние лог. 0
1	1	Выход PA2 устанавливается в состояние лог. 1

5.4. Таймер/счетчик T1

Таймер/счетчик T1 присутствует только в модели ATtiny15L и имеет несколько дополнительных функций по сравнению с таймером/счетчиком T0. В отличие от таймера/счетчика T0, возможность счета внешних импульсов в таймере T1 отсутствует. Однако таймер/счетчик T1 может выполнять определенные действия при равенстве содержимого счетного регистра заданному значению. Кроме того, он может работать как генератор сигнала с ШИМ. Причем генерация сигнала с ШИМ «вынесена» в отдельный режим работы таймера/счетчика, в ко-

тором недоступны остальные функции (кроме генерации прерываний). Режим генерации сигнала с ШИМ будем называть «режим ШИМ», а режим, в котором доступны остальные функции таймера/счетчика — «режим таймера».

Структурная схема таймера/счетчика T1 приведена на **Рис. 1.53**. В его состав входят три 8-разрядных регистра (счетный регистр TCNT1 и два регистра сравнения OCR1A и OCR1B), два 8-разрядных компаратора, два управляющих регистра (регистр управления TCCR1 и регистр специальных функций SFIOR), а также блок управления таймером.

Все флаги состояния таймера/счетчика (переполнения и совпадения) находятся в регистре флагов прерываний от таймеров TIFR. А разрешение и запрещение прерываний от таймера осуществляется установкой/сбросом соответствующих флагов регистра TIMSK.

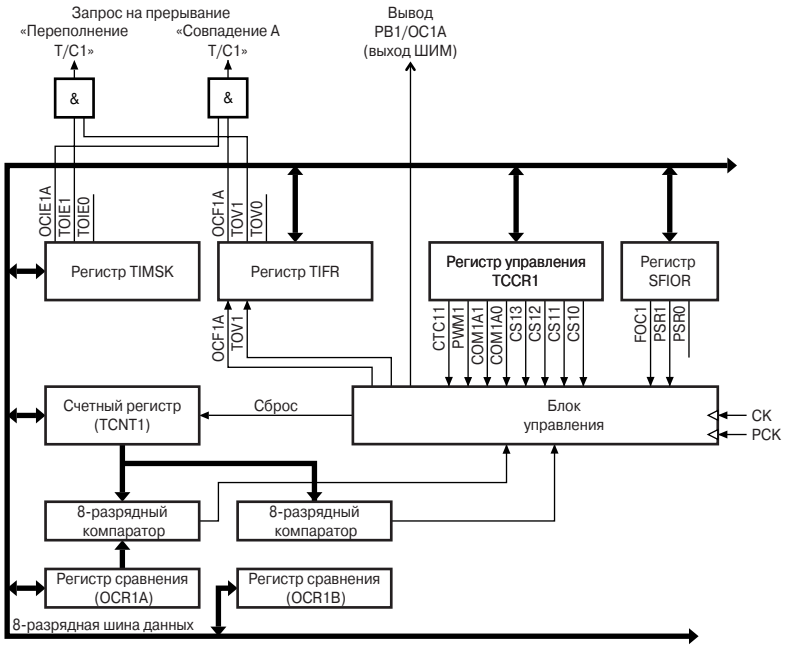


Рис. 1.53. Структурная схема таймера/счетчика T1 (модель ATtiny15L)

Счетный регистр таймера/счетчика TCNT1, расположенный по адресу $2F, реализован как суммирующий счетчик и доступен в любой момент времени как для чтения, так и для записи. При записи в регистр TCNT1 во время работы таймера счет будет продолжен по сле-

дующему за операцией записи импульсу тактового сигнала таймера/счетчика. После подачи напряжения питания в регистре TCNT1 находится нулевое значение.

Управление таймером/счетчиком T1 осуществляется с помощью регистра управления TCCR1 и регистра специальных функций SFIOR, расположенных по адресам $30 и $2C соответственно.

Формат регистра TCCR1 приведен на **Рис. 1.54**. Формат регистра SFIOR приведен на **Рис. 1.55**. Краткое описание функций разрядов этих регистров приведено в **Табл. 1.39** и **Табл. 1.40**, а подробно назначение отдельных разрядов будет описано ниже. Неиспользуемые разряды регистров доступны только для чтения и содержат лог. 0.

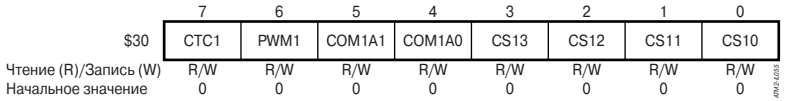


Рис. 1.54. Формат регистра TCCR1

Таблица 1.39. **Разряды регистра TCCR1**

Разряд	Название	Описание
7	CTC1	Сброс таймера/счетчика по совпадению («1» — сбросить при наступлении события)
6	PWM1	Разрешение ШИМ («1» — включить режим ШИМ)
5, 4	COM1A1, COM1A0	Задание режима работы вывода PB1 (OC1A)
3...0	CS13...CS10	Выбор источника тактового сигнала

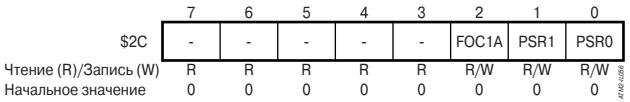


Рис. 1.55. Формат регистра SFIOR

Таблица 1.40. **Разряды регистра SFIOR**

Разряд	Название	Описание
7...3	—	Зарезервированы
2	FOC1A	Принудительное формирование события совпадения («1» — сформировать событие)
1	PSR1	Сброс предделителя таймера/счетчика T1 («1» — сбросить)
0	PSR0	Сброс предделителя таймера/счетчика T0 («1» — сбросить)

5.4.1. Выбор источника тактового сигнала

Выбор источника тактового сигнала, а также запуск и остановка таймера/счетчика осуществляются с помощью разрядов CS13...CS10 регистра управления TCCR1. Соответствие между состоянием этих разрядов и режимом работы таймера/счетчика приведено в **Табл. 1.41**.

Таблица 1.41. **Выбор источника тактового сигнала для таймера/счетчика T1**

Регистр TCCR1B (TCCR1)				Источник тактового сигнала
CS13	CS12	CS11	CS10	
0	0	0	0	Таймер/счетчик остановлен
0	0	0	1	CK × 16 (=PCK)
0	0	1	0	CK × 8 (=PCK/2)
0	0	1	1	CK × 4 (=PCK/4)
0	1	0	0	CK × 2 (=PCK/8)
0	1	0	1	CK (частота тактового сигнала микроконтроллера)
0	1	1	0	CK/2
0	1	1	1	CK/4
1	0	0	0	CK/8
1	0	0	1	CK/16
1	0	1	0	CK/32
1	0	1	1	CK/64
1	1	0	0	CK/128
1	1	0	1	CK/256
1	1	1	0	CK/512
1	1	1	1	CK/1024

Обратите внимание, что в строках 2...5 таблицы указаны значения тактовой частоты таймера/счетчика, превышающие значение тактовой частоты микроконтроллера. Дело в том, что в микроконтроллере ATtiny15L имеется встроенный синтезатор частоты, формирующий сигнал с частотой, в 16 раз превышающей частоту тактового сигнала встроенного *RC*-генератора. Номинальная частота *RC*-генератора равна 1.6 МГц, соответственно частота сигнала на выходе синтезатора частоты равна 25.6 МГц. Разумеется, при подстройке *RC*-генератора эта частота также изменяется.

Предделитель таймера/счетчика T1, формирующий тактовый сигнал заданной частоты, также может быть принудительно сброшен. Для этого необходимо записать лог. 1 в разряд PSR1 регистра SFIOR.

5.4.2. Режим таймера

Принцип работы таймера/счетчика T1 в этом режиме такой же, что и таймера/счетчика T0. По каждому импульсу, поступающему на тактовый вход таймера/счетчика, производится инкремент содержимого счетного регистра TCNT1. При переходе таймера/счетчика из состояния «$FF» в состояние «$00» устанавливается флаг TOV1 регистра TIFR и генерируется запрос на прерывание. Разрешение прерывания осуществляется установкой в «1» разряда TOIE1 регистра TIMSK.

При каждом изменении счетного регистра осуществляется сравнение его содержимого с числом, находящемся в регистре сравнения OCR1A. При совпадении содержимого этих регистров устанавливается флаг OCF1A регистра TIFR и генерируется запрос на прерывание. Разрешение прерывания осуществляется установкой в «1» разряда OCIE1A регистра TIMSK. Помимо генерации прерывания при равенстве счетного регистра и регистра сравнения могут выполняться также сброс таймера/счетчика или изменение состояния вывода PB1 (OC1A) микроконтроллера.

Поведение микроконтроллера, т. е. выполнение или невыполнение указанных действий, определяется состоянием разрядов регистра управления TCCR1. Описание этих разрядов приведено в **Табл. 1.42**.

Таблица 1.42. **Управление работой схемы сравнения таймера/счетчика T1**

<table>
<tr><th>Разряд</th><th>Название</th><th colspan="3">Описание</th></tr>
<tr><td>7</td><td>CTC1</td><td colspan="3">Сброс таймера/счетчика. Если этот разряд установлен в «1», то при совпадении содержимого счетного регистра и регистра сравнения OCR1A производится сброс таймера/счетчика в состояние «$00»</td></tr>
<tr><td rowspan="6">5, 4</td><td rowspan="6">COM1A1,
COM1A0</td><td colspan="3">Управление выводом PB1 (OC1A)*. Состояние этих разрядов определяет поведение вывода PB1 (OC1A) при совпадении содержимого счетного регистра и регистра сравнения OCR1A. При изменении состояния этих разрядов соответствующее прерывание от компаратора таймера/счетчика рекомендуется запретить (во избежание ложной генерации прерывания). Чтобы таймер/счетчик мог управлять этим выводом, последний должен быть сконфигурирован как выходной. Поведение вывода задается следующим образом:</td></tr>
<tr><td>COM1A1</td><td>COM1A0</td><td>Описание</td></tr>
<tr><td>0</td><td>0</td><td>Таймер/счетчик отключен от вывода PB1</td></tr>
<tr><td>0</td><td>1</td><td>Состояние вывода меняется на противоположное</td></tr>
<tr><td>1</td><td>0</td><td>Вывод сбрасывается в «0»</td></tr>
<tr><td>1</td><td>1</td><td>Вывод устанавливается в «1»</td></tr>
<tr><td colspan="5">* В режиме ШИМ эти разряды имеют другие функции (Табл. 1.43)</td></tr>
</table>

Состояние вывода PB1 может быть изменено и принудительно. Для этого предназначен 2-й разряд регистра SFIOR — FOC1A. При записи в него лог. 1 состояние вывода PB1 (OC1A) изменяется в соответствии с содержимым разрядов COM1A1: COM1A0 регистра TCCR1A (см. **Табл. 1.42**). Прерывание при этом не генерируется. При чтении разряда FOC1A всегда возвращается лог. 0.

5.4.3. Режим ШИМ

В этом режиме таймер/счетчик T1 представляет собой широтно-импульсный модулятор и используется для генерирования сигнала с программируемыми частотой и скважностью. Для перевода таймера/счетчика T1 в режим ШИМ необходимо установить в «1» разряд PWM1 регистра управления таймером TCCR1.

При работе таймера/счетчика T1 в режиме ШИМ состояние счетного регистра изменяется от $00 до значения, находящегося в регистре OCR1B, после чего счетный регистр сбрасывается и цикл повторяется. При равенстве содержимого счетного регистра и регистра сравнения OCR1A состояние вывода PB1 (OC1A) микроконтроллера изменяется в соответствии со значением разрядов COM1A1:COM1A0 регистра TCCR1 (см. **Табл. 1.43** и **Рис. 1.56**). Таким образом, содержимое регистра OCR1A определяет скважность ШИМ-сигнала, а содержимое регистра OCR1B — его частоту (**Табл. 1.44**).

Таблица 1.43. **Поведение вывода PB1 (OC1A) в режиме ШИМ**

COM1A1	COM1A0	Поведение вывода PB1 (OC1A)
0	0	Таймер/счетчик T1 отключен от вывода
0	1	Таймер/счетчик T1 отключен от вывода
1	0	Сбрасывается в «0» при равенстве регистров TCNT1 и OCR1A. Устанавливается в «1» при TCNT1 = $00 (неинвертированный ШИМ-сигнал)
1	1	Устанавливается в «1» при равенстве регистров TCNT1 и OCR1A. Сбрасывается в «0» при TCNT1 = $00 (инвертированный ШИМ-сигнал)

Таблица 1.44. **Зависимость частоты ШИМ-сигнала от тактовой частоты таймера/счетчика T1**

Частота тактового сигнала таймера/счетчика	Содержимое регистра OCR1B	Частота ШИМ-сигнала (кГц)
CK	159	10
PCK/8	159	20
PCK/4	213	30
PCK/4	159	40
PCK/2	255	50
PCK/2	213	60
PCK/2	181	70

Продолжение таблицы 1.44

Частота тактового сигнала таймера/счетчика	Содержимое регистра OCR1B	Частота ШИМ-сигнала (кГц)
PCK/2	159	80
PCK/2	141	90
PCK	255	100
PCK	231	110
PCK	213	120
PCK	195	130
PCK	181	140
PCK	169	150

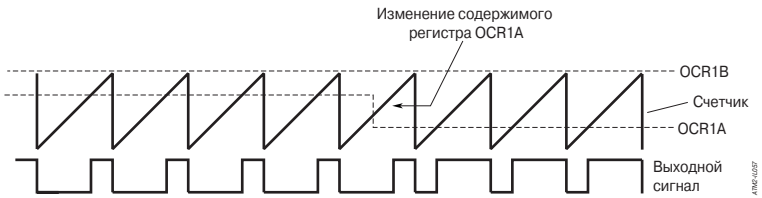


Рис. 1.56. Формирование ШИМ-сигнала

Соответственно, если содержимое регистра сравнения OCR1A равно $00 или содержимому регистра OCR1B, то при следующем совпадении состояния счетчика и содержимого регистра сравнения выход схемы сравнения переключится в устойчивое состояние согласно **Табл. 1.45**.

Таблица 1.45. **Устойчивые состояния вывода PB1 (OC1A)**

COM1A1	COM1A0	Регистр OCR1A	Состояние вывода PB1 (OC1A)
1	0	$000	0
1	0	OCR1B	1
1	1	$000	1
1	1	OCR1B	0

В схеме ШИМ приняты меры, позволяющие избежать появления несимметричных выбросов (glitches). Для этого изменение содержимого регистра OCR1A происходит только в момент достижения счетчиком максимального значения. До этого момента записываемое в регистр OCR1A число сохраняется в специальном временном регистре. Соответственно при чтении регистра сравнения в промежутке между записью в него и его действительным изменением возвращается содержимое временного регистра. Таким

образом, при чтении регистра всегда возвращается значение, записанное последним.

И напоследок несколько слов о прерываниях. При работе таймера/счетчика T1 в режиме ШИМ может генерироваться прерывание по переполнению счетного регистра таймера/счетчика, а также прерывание от схемы сравнения. Разрешение и обработка соответствующих прерываний выполняется как обычно.

5.5. Сторожевой таймер

Основная функция сторожевого таймера — защита устройства от сбоев. Благодаря сторожевому таймеру можно прервать выполнение зациклившейся программы или выйти из других непредвиденных ситуаций, препятствующих нормальному выполнению программы.

Структурная схема сторожевого таймера приведена на **Рис. 1.57**.

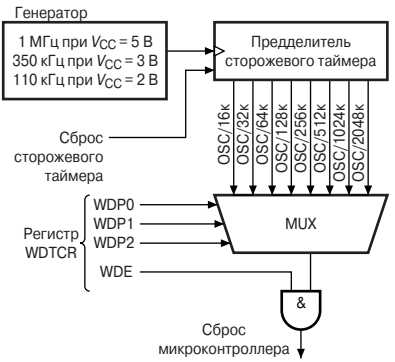


Рис. 1.57. Структурная схема сторожевого таймера

Сторожевой таймер имеет независимый тактовый генератор и работает даже в режиме Power Down. Частота этого генератора зависит от напряжения питания устройства, температуры, технологического разброса. Типовые значения частот равны: 1 МГц при V_{CC} = 5.0 В, 350 кГц при V_{CC} = 3.0 В и 110 кГц при V_{CC} = 2.0 В.

Если сторожевой таймер включен, то через определенные промежутки времени (при наступлении тайм-аута) выполняется сброс микроконтроллера. Подробнее о процессе сброса см. в разделе 3.4. При нормальном выполнении программы сторожевой таймер должен пе-

риодически (через промежутки времени, меньшие его периода) сбрасываться командой WDR.

Для управления сторожевым таймером предназначен регистр WDTCR (Watch Dog Timer Control Register), расположенный по адресу $21 (в моделях ATtiny28x — по адресу $01). Формат этого регистра приведен на **Рис. 1.58**, а краткое описание разрядов этого регистра приведено в **Табл. 1.46**.

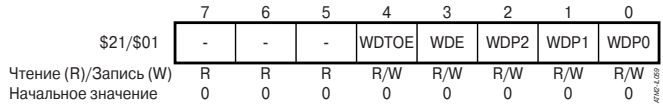


Рис. 1.58. Формат регистра WDTCR

Таблица 1.46. **Разряды регистра WDTCR**

Разряд	Название	Описание
7...5	—	Зарезервированы
4	WDTOE	Разрешение выключения сторожевого таймера
3	WDE	Разрешение сторожевого таймера («1» – включен)
2...0	WDP2...WDP0	Коэффициент деления предделителя сторожевого таймера

Для включения/выключения сторожевого таймера используются два разряда регистра WDTCR — WDE и WDTOE. Если разряд WDE (Watch Dog Enable) установлен в «1», сторожевой таймер включен, если разряд WDE сброшен в «0», сторожевой таймер выключен. Непосредственно перед включением таймера рекомендуется также выполнять его сброс командой WDR.

Чтобы избежать непреднамеренного выключения сторожевого таймера, предназначен разряд WDTOE (Watch Dog Timer On Enable). Дело в том, что выключение сторожевого таймера (сброс разряда WDE) можно осуществить только при установленном разряде WDTOE, который аппаратно сбрасывается через 4 машинных цикла после установки в «1». За счет этого практически исключается возможность случайного выключения сторожевого таймера.

Исходя из сказанного, для выключения сторожевого таймера необходимо одной командой записать лог. 1 в разряды WDE и WDTOE или в течение следующих четырех машинных циклов записать лог. 0 в разряд WDE.

Период наступления тайм-аута сторожевого таймера задается с помощью разрядов WDP2...WDP0 регистра WDTCR согласно **Табл. 1.47**.

Таблица 1.47. **Задание периода сторожевого таймера**

WDP2	WDP1	WDP0	Число тактов генератора	Период наступления тайм-аута (типовое значение)		
				V_{CC} = 2.0 В	V_{CC} = 3.0 В	V_{CC} = 5.0 В
0	0	0	16K	0.15 мс	47 мс	15 мс
0	0	1	32K	0.30 мс	94 мс	30 мс
0	1	0	64K	0.60 мс	0.19 с	60 мс
0	1	1	128K	1.2 с	0.38 с	0.12 с
1	0	0	256K	2.4 с	0.75 с	0.24 с
1	0	1	512K	4.8 с	1.5 с	0.49 с
1	1	0	1024K	9.6 с	3.0 с	0.97 с
1	1	1	2048K	19 с	6.0 с	1.9 с

Чтобы избежать непреднамеренного сброса микроконтроллера при изменении периода сторожевого таймера, необходимо перед записью разрядов WDP2...WDP0 либо запретить сторожевой таймер, либо сбросить его.

Глава 6. **Аналоговый компаратор**

6.1. Общие сведения

Аналоговый компаратор входит в состав всех без исключения моделей микроконтроллеров семейства. Будучи включенным, этот компаратор позволяет сравнивать значения напряжений, присутствующих на двух выводах микроконтроллера. Результатом сравнения является логическое значение, которое может быть прочитано из программы. По результату сравнения также может быть сгенерировано прерывание.

Используемые компаратором выводы являются контактами портов ввода/вывода общего назначения. В качестве неинвертирующего входа (AIN0) используется вывод PB0, а в качестве инвертирующего (AIN1) — PB1.

Чтобы указанные линии портов ввода/вывода могли использоваться аналоговым компаратором, они должны быть сконфигурированы как входы (соответствующий разряд регистра DDRB установлен в «1»). Внутренние подтягивающие резисторы, если они подключены, при разрешении работы компаратора отключаются автоматически.

6.2. Функционирование компаратора

Структурная схема аналогового компаратора приведена на **Рис. 1.59**. Обратите внимание на то, что источник опорного напряжения и связанный с ним мультиплексор присутствуют только в моделях ATtiny12x и ATtiny15L.

Управление компаратором и контроль его состояния осуществляется с помощью регистра ACSR (Analog Comparator Status Register), расположенного по адресу $08. Формат этого регистра для различных моделей микроконтроллеров приведен на **Рис. 1.60**. Краткое описание разрядов этого регистра приведено в **Табл. 1.48**.

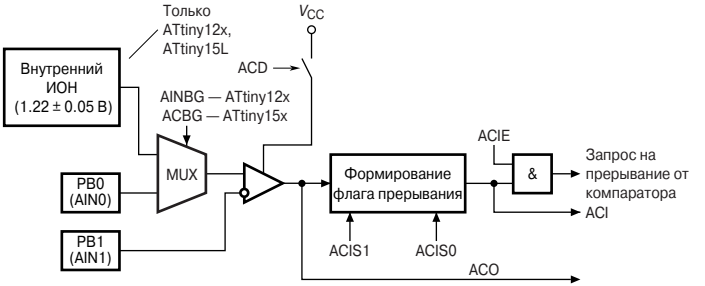


Рис. 1.59. Структурная схема аналогового компаратора

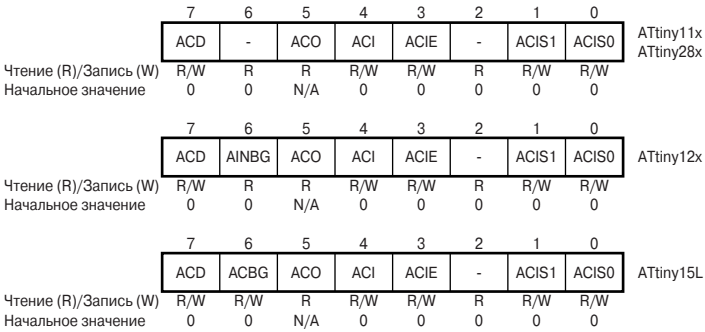


Рис. 1.60. Формат регистра ACSR

Таблица 1.48. **Разряды регистра ACSR**

Разряд	Название	Описание	Модель
7	ACD	Выключение компаратора («0» — включен, «1» — выключен)	Все
6	AINBG	Подключение к неинвертирующему входу компаратора внутреннего ИОН («0» — не подключен, «1» — подключен)	ATtiny12x
	ACBG	То же	ATtiny15L
	—	Зарезервирован	ATtiny11x ATtiny28x
5	ACO	Результат сравнения (выход компаратора)	Все
4	ACI	Флаг прерывания от компаратора	Все
3	ACIE	Разрешение прерывания от компаратора	Все
2	—	Зарезервирован	Все
1, 0	ACIS1:ACIS0	Условие возникновения прерывания от компаратора	Все

По своему действию рассматриваемый узел микроконтроллера является обычным компаратором. Если напряжение на выводе AIN0 (неинвертирующий вход) больше напряжения на выводе AIN1(инвертирующий вход), то результат сравнения будет равен «1». В противном случае результат сравнения будет равен «0». Этот результат сохраняется в разряде ACO регистра ACSR.

Разряд ACD отвечает за включение и выключение компаратора. При включении напряжения питания все разряды регистра ACSR сбрасываются в «0», поэтому компаратор автоматически включается при подаче напряжения питания на микроконтроллер. Чтобы его выключить, разряд ACD следует установить в «1». При изменении состояния этого разряда необходимо запретить прерывание от компаратора.

Как уже было отмечено, в соответствии с результатом сравнения схема компаратора может генерировать запрос на прерывание. Если состояние выхода компаратора (разряд ACO) изменилось заданным образом, устанавливается флаг прерывания ACI регистра ACSR и генерируется запрос на прерывание. Как и для других прерываний, этот флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания или программно, записью в него лог. 1. Для разрешения прерывания необходимо установить в «1» разряд ACIE регистра ACSR и, разумеется, флаг I регистра SREG.

Условие генерации запроса на прерывание от компаратора определяется состоянием разрядов ACIS1:ACIS0 регистра ACSR в соответствии с **Табл. 1.49**. При изменении этих разрядов прерывание от компаратора (как и для разряда ACD) должно быть запрещено.

Таблица 1.49. **Условия генерации запроса на прерывание от компаратора**

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

В микроконтроллерах ATtiny12x и ATtiny15L к неинвертирующему входу компаратора вместо вывода AIN0 микроконтроллера может быть подключен внутренний источник опорного напряжения величиной

1.22 ± 0.05 В. Для этого необходимо установить в «1» разряд AINBG (ACBG для ATtiny15L) регистра ACSR.

И в заключение в **Табл. 1.50** представлены основные параметры аналогового компаратора.

Таблица 1.50. **Параметры аналогового компаратора**

Обозначение	Параметр	Условия	min	max
V_{ACIO}	Входное напряжение смещения [мВ]	V_{CC} = 5 В, $V_{IN} = V_{CC}/2$	—	40.0
I_{ACLK}	Ток утечки на входе [нА]		–50.0	50.0
t_{ACPD}	Время отклика [нс]	V_{CC} = 2.7 В V_{CC} = 4.0 В	—	750 500

Глава 7. **Аналого-цифровой преобразователь**

7.1. Общие сведения

В состав микроконтроллера ATtiny15L входит 10-разрядный АЦП последовательного приближения. Этот АЦП имеет следующие основные параметры:

- абсолютная погрешность: ±2 МЗР;
- интегральная нелинейность: ±0.5 МЗР;
- быстродействие: до 15 тыс. выборок/с.

На входе собственно АЦП располагается 4-канальный аналоговый мультиплексор, предоставляющий в распоряжение пользователя 4 канала с несимметричными входами либо 1 канал с дифференциальным входом с возможностью 20-кратного предварительного усиления. Для несимметричных входов диапазон входных напряжений составляет 0...V_{CC}, а для дифференциального входа – 0...2.6 В. В качестве источника опорного напряжения ИОН для АЦП может использоваться внутренний или внешний источник опорного напряжения или напряжение питания микроконтроллера.

В процессе работы АЦП может функционировать в двух режимах:

- режим одиночного преобразования — запуск каждого преобразования инициируется пользователем;
- режим непрерывного преобразования — запуск преобразований выполняется непрерывно через определенные интервалы времени.

7.2. Функционирование модуля АЦП

Структурная схема модуля АЦП приведена на **Рис.1.61**.

Управление модулем АЦП и контроль его состояния осуществляется с помощью регистра ADCSR (Analog Digital Converter Status Register — регистр состояния АЦП), расположенного по адресу $06. Формат этого

регистра приведен на **Рис. 1.62**, а краткое описание функций разрядов регистра приведено в **Табл. 1.51**. Подробно использование различных разрядов регистра будет описано далее.

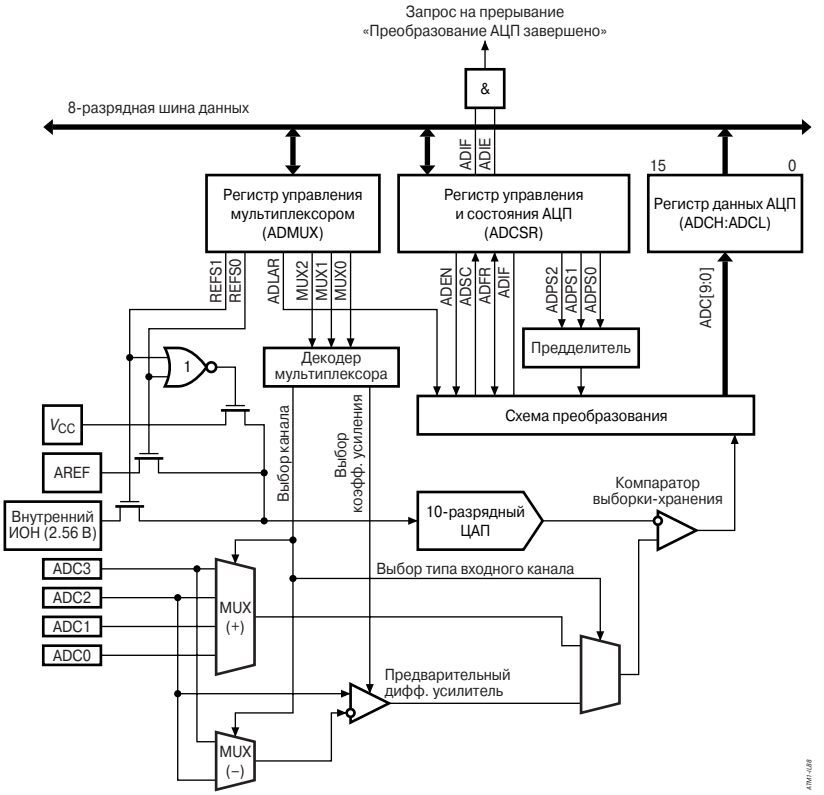


Рис. 1.61. Структурная схема модуля АЦП

	7	6	5	4	3	2	1	0
$06	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Чтение (R)/Запись (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	0	0	0	0

Рис. 1.62. Формат регистра ADCSR

Таблица 1.51. **Разряды регистра ADCSR**

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

Для разрешения работы АЦП необходимо записать лог. 1 в разряд ADEN регистра ADCSR, а для выключения соответственно лог. 0. Причем, если АЦП будет выключено во время цикла преобразования, то преобразование завершено не будет (в регистре данных АЦП останется результат предыдущего преобразования).

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

Запуск каждого преобразования в режиме одиночного преобразования, а также запуск первого преобразования в режиме непрерывного преобразования осуществляется установкой в «1» разряда ADSC регистра ADCSR. Собственно цикл преобразования начинается по первому нарастающему фронту тактового сигнала после установки этого разряда. Как правило, длительность цикла составляет 13 тактов; выборка и запоминание входного сигнала осуществляется в течение первых 1.5 тактов. Через 13 тактов преобразование завершается, разряд ADSC аппаратно сбрасывается в «0» (в режиме одиночного преобразования) и результат преобразования сохраняется в регистре данных АЦП. Одновременно устанавливается флаг прерывания ADIF регистра ADCSR и генерируется запрос на прерывание. Как и флаги остальных прерываний, флаг ADIF сбрасывается аппаратно при запуске подпрограммы обработки прерывания от АЦП или программно, записью в него лог. 1. Разрешение прерывания осуществляется установкой в «1» разряда ADIE регистра ADCSR при установленном флаге I регистра SREG.

Если АЦП работает в режиме непрерывного преобразования, новый цикл начнется сразу же после записи результата. В режиме одиночного преобразования новое преобразование может быть запущено

сразу же после сброса разряда ADSC (до сохранения результата текущего преобразования). Однако реально цикл преобразования начнется не ранее чем через один такт после окончания текущего преобразования. Временные диаграммы, иллюстрирующие сказанное, приведены на **Рис. 1.63**.

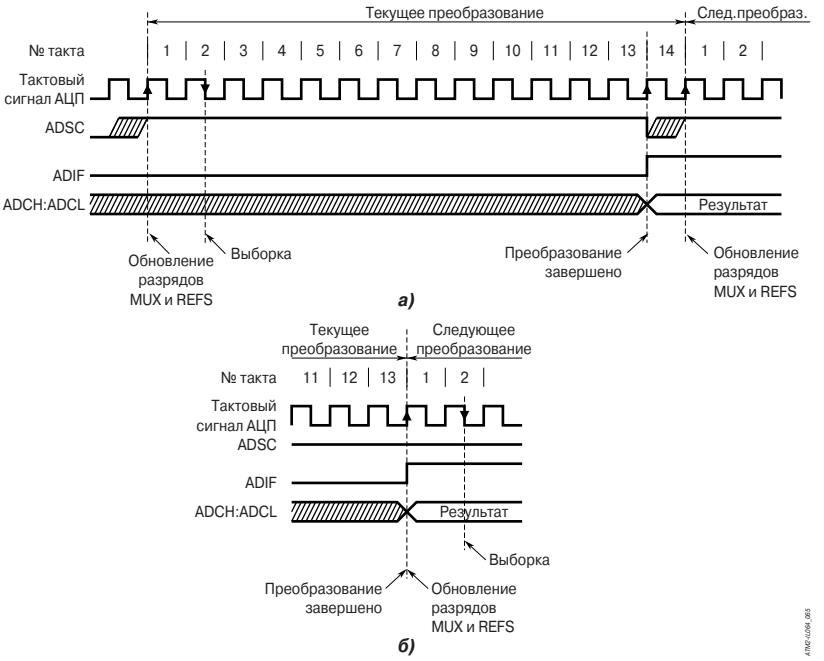


Рис. 1.63. Режим одиночного (*а*) и непрерывного (*б*) преобразования АЦП

При использовании АЦП следует учитывать, что в ряде случаев для выполнения преобразования может потребоваться 25 тактов, т. е. на 12 тактов больше, чем обычно. Это происходит при запуске первого преобразования после наступления следующих событий:

- включение АЦП;
- смена источника опорного напряжения;
- включение канала с дифференциальным входом.

В течение этих 12 тактов выполняется «холостое» преобразование, инициализирующее АЦП (**Рис. 1.64**).

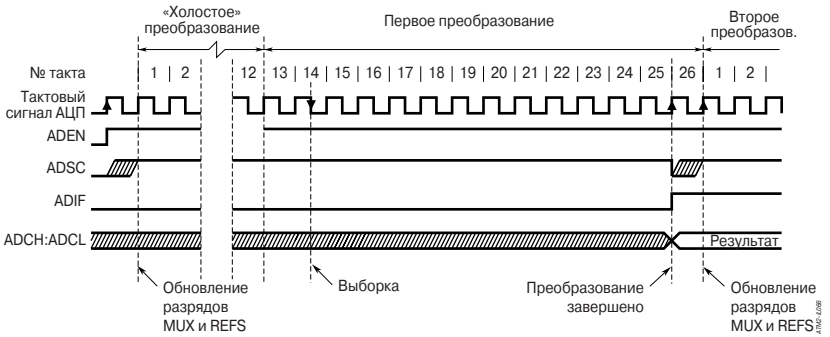


Рис. 1.64. Временные диаграммы работы АЦП при первом преобразовании (режим одиночного преобразования)

Тактовым сигналом модуля АЦП является сигнал с предделителя, на вход которого, в свою очередь, поступает тактовый сигнал микроконтроллера. Коэффициент деления предделителя и соответственно длительность преобразования определяется состоянием разрядов ADPS2...ADPS0 регистра ADCSR (**Табл. 1.52**).

Таблица 1.52. **Задание коэффициента деления предделителя АЦП**

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

Наибольшая точность преобразования достигается, если тактовая частота модуля АЦП находится в диапазоне 50...200 кГц. Соответственно, коэффициент деления предделителя рекомендуется выбирать таким, чтобы тактовая частота модуля АЦП находилась в указанном диапазоне.

Как уже было сказано, результат преобразования сохраняется в регистре данных АЦП. Поскольку АЦП 10-разрядное, этот регистр физически размещен в двух регистрах ввода/вывода, доступных только для чтения:

ADCH и ADCL. Эти регистры при включении микроконтроллера содержат значение «$0000» и расположены по адресам $05 и $04. Обращение к этим регистрам (для получения результата преобразования) должно выполняться в определенной последовательности: сначала необходимо прочитать регистр ADCL, а затем ADCH. Это требование связано с тем, что после обращения к регистру ADCL процессор блокирует доступ к регистрам данных со стороны АЦП до тех пор, пока не будет прочитан регистр ADCH. Благодаря этому можно быть уверенным, что при чтении регистров в них будут находиться составляющие одного и того же результата. Соответственно, если очередное преобразование завершится до обращения к регистру ADCH, результат преобразования будет потерян. С другой стороны, если для результата преобразования достаточно точности 8 разрядов, то для его получения достаточно прочитать содержимое регистра ADCH.

Для управления выравниванием результата преобразования служит разряд ADLAR регистра ADMUX (см. ниже). Если этот разряд установлен в «1», результат преобразования выравнивается по левой границе 16-разрядного слова, если сброшен в «0» — по правой границе.

Управление входным мультиплексором модуля АЦП, а также дополнительное управление модулем АЦП осуществляется с помощью регистра ADMUX, расположенного по адресу $07. Формат этого регистра приведен на **Рис. 1.65**, а краткое описание функций разрядов регистра приведено в **Табл. 1.53**.

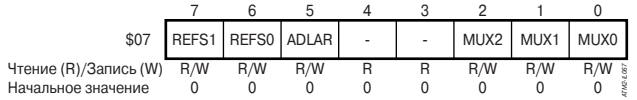


Рис. 1.65. Формат регистра ADMUX

Таблица 1.53. **Разряды регистра ADMUX**

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

Разряды MUX2...MUX0 этого регистра определяют номер активного канала (аналоговый вход, подключенный к входу АЦП) согласно **Табл. 1.54**. При выборе канала с дифференциальным входом эти разряды определяют также коэффициент предварительного усиления входного сигнала.

Таблица 1.54. **Управление входным мультиплексором**

MUX2...0	Несимметричный вход	Дифференциальный вход (положительный)	Дифференциальный вход (отрицательный)	Предв. усиление
000	ADC0 (PB5)	Не применимо		
001	ADC1 (PB2)			
010	ADC2 (PB3)			
011	ADC3 (PB4)			
100*	Не применимо	ADC2 (PB3)	ADC2 (PB3)	1x
101*		ADC2 (PB3)	ADC2 (PB3)	20x
110		ADC2 (PB3)	ADC3 (PB4)	1x
111		ADC2 (PB3)	ADC3 (PB4)	20x
* Используется только для калибровки смещения.				

Обратите внимание, что при выборе канала с дифференциальным входом оба входа дифференциального усилителя можно подключить к одному и тому же выводу микроконтроллера ADC2 (PB3). Результат преобразования в этом случае будет равен величине суммарного смещения входного усилителя и схемы преобразования. Это значение впоследствии может быть вычтено из результата последующих преобразований (при том же коэффициенте усиления) для снижения ошибки смещения до величины, меньшей младшего значащего разряда (МЗР).

Состояние разрядов MUX2...MUX0 можно изменить в любой момент, однако, если это будет сделано во время цикла преобразования, смена канала произойдет только после завершения преобразования. Благодаря этому в режиме непрерывного преобразования можно легко реализовать сканирование каналов. Под этим термином в данном случае понимается последовательное (циклическое или по заданной программе) преобразование сигналов нескольких каналов.

Как уже было отмечено, модуль АЦП может использовать различные источники опорного напряжения. Выбор конкретного источника опорного напряжения осуществляется с помощью разрядов REFS1 и REFS0 регистра ADMUX (**Табл. 1.55**).

В последнем случае (REFS1:REFS0 = «11») к выводу PB0 (AREF) можно подключить внешний фильтрующий конденсатор для повышения помехозащищенности.

Таблица 1.55. **Выбор источника опорного напряжения**

REFS1	REFS0	Источник опорного напряжения
0	0	Напряжение питания микроконтроллера (V_{CC}); только для каналов с несимметричными входами
0	1	Внешний ИОН, подключенный к выводу PB0 (AREF); внутренний ИОН отключен
1	0	Внутренний ИОН напряжением 2.56 В, отключенный от вывода PB0 (AREF)
1	1	Внутренний ИОН напряжением 2.56 В, подключенный к выводу PB0 (AREF)

7.3. Повышение точности преобразования

Как уже было сказано, для минимизации погрешности самого АЦП необходимо правильно выбрать тактовую частоту модуля. Вторым фактором, влияющим на точность преобразования, являются различного рода помехи и шумы.

Известно, что работающий микроконтроллер является источником электромагнитных помех. Чтобы свести к минимуму помехи, наводимые ядром процессора, в микроконтроллере предусмотрен специальный «спящий» режим — ADC Noise Reduction (режим снижения шумов АЦП). В этом режиме из всех периферийных устройств функционируют только АЦП и сторожевой таймер. Для той же цели (но с меньшим эффектом) может быть использован режим Idle. Для использования АЦП в «спящем» режиме необходимо убедиться, что АЦП включено и не занято преобразованием. Затем переключить АЦП в режим одиночного преобразования и разрешить прерывание от АЦП, после чего перевести микроконтроллер в режим ADC Noise Reduction (или режим Idle).

Сразу же после остановки процессора начнется цикл преобразования. При завершении преобразования будет сгенерировано прерывание от АЦП, которое переведет микроконтроллер в рабочий режим и начнется выполнение подпрограммы обработки этого прерывания. Однако помехи генерируются не только ядром процессора, но и другими схемами, в том числе расположенными вне микроконтроллера. Для уменьшения этих помех при разработке конструкции и разводке печатной платы рекомендуется придерживаться следующих правил:

- на печатной плате необходимо предусмотреть область (или даже слой) сплошной металлизации под аналоговую «землю». Аналоговая часть микроконтроллера и аналоговая часть всего устройства должны располагаться над этой областью. Аналоговая и цифровая «земли» должны соединяться друг с другом в единственной точке печатной платы;
- проводники, по которым распространяются аналоговые сигналы, должны быть как можно короче и располагаться над аналоговой «землей». Кроме того, они должны быть размещены как можно дальше от быстродействующих цифровых цепей.

7.4. Параметры АЦП

Основные параметры АЦП приведены в **Табл. 1.56**. Все значения указаны для диапазона температур окружающей среды –40...+80°C.

Таблица 1.56. **Основные параметры АЦП**

Обозначение	Параметр	Условия	min	typ	max
—	Разрешение [бит]	Несимметричный вход	—	10	—
		Дифференциальный вход	—	8	—
—	Абсолютная погрешность [МЗР[	V_{REF} = 4 В, f_{ADC} = 200 кГц	—	1	2
		V_{REF} = 4 В, f_{ADC} = 1 МГц	—	4	—
		V_{REF} = 4 В, f_{ADC} = 2 МГц	—	16	—
		AV_{CC} = 3.3...6.0 В	—	—	2
INL	Интегральная нелинейность [МЗР]	V_{REF} > 2 В	—	0.5	—
DNL	Дифференциальная нелинейность [МЗР]	V_{REF} > 2 В	—	0.5	—
—	Ошибка смещения [МЗР]	V_{REF} > 2 В	—	1	—
—	Время преобразования [мкс]	Режим непрерывного преобразования	65	—	260
f_{ADC}	Тактовая частота [кГц]	—	50	—	200
V_{REF}	Опорное напряжение [В]	Несимметричный вход	2.0	—	V_{CC}
		Дифференциальный вход	2.0	—	V_{CC} – – 0.2
V_{INT}	Напряжение внутреннего ИОН [В]	—	2.4	2.56	2.7
R_{REF}	Входное сопротивление канала опорного напряжения [кОм]	—	6	10	13
R_{AIN}	Входное сопротивление аналогового входа [МОм]	—	—	100	—

Часть 2
Микроконтроллеры семейства Mega

Глава 8. **Знакомство с семейством Mega**

8.1. Общие сведения

Как и все микроконтроллеры AVR фирмы «Atmel», микроконтроллеры семейства Mega являются 8-разрядными микроконтроллерами, предназначенными для встраиваемых приложений. Они изготавливаются по малопотребляющей КМОП-технологии, которая в сочетании с усовершенствованной RISC-архитектурой позволяет достичь наилучшего соотношения быстродействие/энергопотребление. Микроконтроллеры описываемого семейства являются наиболее развитыми представителями микроконтроллеров AVR.

8.2. Отличительные особенности

К числу особенностей микроконтроллеров AVR семейства Mega относятся:

- FLASH-память программ объемом 8...128 Кбайт (число циклов стирания/записи не менее 1000);
- оперативная память (статическое ОЗУ) объемом 1...4 Кбайт;
- память данных на основе ЭСППЗУ (EEPROM) объемом 512 байт...4 Кбайт (число циклов стирания/записи не менее 100000);
- 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);

- различные способы синхронизации: встроенный *RC*-генератор с внутренней или внешней времязадающей *RC*-цепочкой или с внешним резонатором (пьезокерамическим или кварцевым); внешний сигнал синхронизации;
- наличие нескольких режимов пониженного энергопотребления;
- наличие детектора снижения напряжения питания (brown-out detector, BOD);
- возможность программного снижения частоты тактового генератора*.

8.3. Характеристики процессора

Подавляющее большинство основных характеристик процессора микроконтроллеров семейства Mega такие же, что и у микроконтроллеров других семейств — Classic и Tiny:

- полностью статическая архитектура; минимальная тактовая частота равна нулю;
- АЛУ подключено непосредственно к регистрам общего назначения;
- большинство команд выполняются за один машинный цикл;
- многоуровневая система прерываний; поддержка очереди прерываний.

В то же время процессор микроконтроллеров семейства Mega имеет ряд характеристик, присущих именно этому семейству:

- наибольшее число источников прерываний (до 27 источников, из них до 8 внешних);
- наличие программного стека во всех моделях семейства;
- наличие аппаратного умножителя.

8.4. Характеристики подсистемы ввода/вывода

Все характеристики подсистемы ввода/вывода микроконтроллеров семейства Mega такие же, как и у микроконтроллеров других семейств:

- программное конфигурирование и выбор портов ввода/вывода;
- выводы могут быть запрограммированы как входные или как выходные независимо друг от друга;
- входные буферы с триггером Шмитта на всех выводах;
- возможность подключения ко всем входам внутренних подтягивающих резисторов (сопротивление резисторов составляет 35...120 кОм).

* Не во всех моделях.

8.5. Периферийные устройства

Микроконтроллеры семейства Mega имеют наиболее богатый набор периферийных устройств (ПУ). При этом в большинстве моделей имеются все ПУ, которые вообще встречаются в составе микроконтроллеров AVR. Этими устройствами являются:

- 8-разрядные таймеры/счетчики (таймеры T0 и T2). В ряде моделей эти таймеры/счетчики могут работать в качестве часов реального времени (в асинхронном режиме);
- 16-разрядные таймеры/счетчики (таймеры T1 и T3);
- сторожевой таймер WDT;
- генераторы сигнала с ШИМ разрядностью 8 бит (один из режимов работы 8-разрядных таймеров/счетчиков T0 и T2);
- одно-, двух- и трехканальные генераторы сигнала с ШИМ регулируемой разрядности (один из режимов работы 16-разрядных таймеров T1 и T3). Разрешение ШИМ-сигнала для разных моделей составляет 8...10 бит или 1...16 бит;
- аналоговый компаратор;
- многоканальный 10-разрядный АЦП как с несимметричными, так и с дифференциальными входами;
- полнодуплексный универсальный асинхронный приемопередатчик (UART);
- полнодуплексный универсальный синхронный/асинхронный приемопередатчик (USART);
- последовательный синхронный интерфейс SPI;
- последовательный двухпроводный интерфейс TWI (аналог интерфейса I^2C).

8.6. Архитектура ядра

Ядро микроконтроллеров AVR семейства Mega, как и ядро микроконтроллеров семейств Classic и Tiny, выполнено по усовершенствованной RISC-архитектуре (enhanced RISC). Арифметико-логическое устройство (АЛУ), выполняющее все вычисления, подключено непосредственно к 32-м рабочим регистрам, объединенным в регистровый файл. Благодаря этому АЛУ выполняет одну операцию (чтение содержимого регистров, выполнение операции и запись результата обратно в регистровый файл) за один машинный цикл. Практически каждая из команд (за исключением команд, у которых одним из операндов является 16-разрядный адрес) занимает одну ячейку памяти программ.

В микроконтроллерах AVR реализована Гарвардская архитектура, которая характеризуется раздельной памятью программ и данных, каждая из которых имеет собственные шины доступа к ним. Такая организация позволяет одновременно работать как с памятью программ, так и с памятью данных. Разделение шин доступа позволяет использовать для каждого типа памяти шины различной разрядности, причем способы адресации и доступа к каждому типу памяти также различны.

Еще одним решением, направленным на повышение быстродействия, является использование технологии конвейеризации. Конвейеризация заключается в том, что во время исполнения текущей команды производится выборка из памяти и дешифрация кода следующей команды. Причем, поскольку длительность машинного цикла микроконтроллеров AVR составляет всего один период тактового генератора, они могут обеспечивать ту же производительность, что и RISC-микроконтроллеры других фирм, но при более низкой тактовой частоте.

8.7. Цоколевка и описание выводов

В семейство Mega на сегодняшний день входит в общей сложности 16 моделей микроконтроллеров:

- ATmega8, ATmega8L (**Рис. 2.1**) имеют FLASH-память программ объемом 8 Кбайт, ОЗУ объемом 1 Кбайт и EEPROM-память данных объемом 512 байт. Максимальное количество контактов ввода/вывода равно 23;
- ATmega8515, ATmega8515L (**Рис. 2.2**) имеют FLASH-память программ объемом 8 Кбайт, ОЗУ объемом 512 байт (с возможностью подключения внешнего ОЗУ объемом до 64 Кбайт) и EEPROM-память данных объемом 512 байт. Максимальное количество контактов ввода/вывода равно 35;
- ATmega16, ATmega16L (**Рис. 2.3**) имеют FLASH-память программ объемом 16 Кбайт, ОЗУ объемом 1 Кбайт и EEPROM-память данных объемом 512 байт. Максимальное количество контактов ввода/вывода равно 32. Эти модели полностью (функционально и по цоколевке) совместимы с микроконтроллерами ATmega163(L) и предназначены для замены их в новых разработках;
- ATmega161, ATmega161L (**Рис. 2.4**) имеют FLASH-память программ объемом 16 Кбайт, ОЗУ объемом 1 Кбайт и EEPROM-память данных объемом 512 байт. Максимальное количество контактов ввода/вывода равно 35;
- ATmega162, ATmega162L (**Рис. 2.5**) имеют FLASH-память программ объемом 16 Кбайт, ОЗУ объемом 1 Кбайт (с возможностью подключения внешнего ОЗУ объемом до 64 Кбайт) и

EEPROM-память данных объемом 512 байт. Максимальное количество контактов ввода/вывода равно 35. Эти модели по цоколевке полностью совместимы с моделями ATmega161x. Кроме того, они могут работать в режиме совместимости с моделями ATmega161x, обеспечивающем также функциональную совместимость;

- ATmega163, ATmega163L (**Рис. 2.6**) имеют FLASH-память программ объемом 16 Кбайт, ОЗУ объемом 1 Кбайт и EEPROM-память данных объемом 512 байт. Максимальное количество контактов ввода/вывода равно 32. В то время когда писалась книга, этим кристаллам был присвоен статус «не рекомендовано для новых разработок». Вместо них фирма «Atmel» предлагает использовать микроконтроллеры ATmega16x;
- ATmega323, ATmega323L (**Рис. 2.7**) имеют FLASH-память программ объемом 32 Кбайт, ОЗУ объемом 2 Кбайт и EEPROM-память данных объемом 1 Кбайт. Максимальное количество контактов ввода/вывода равно 32. К моменту окончания работы над книгой, этим кристаллам был присвоен статус «не рекомендовано для новых разработок». Вместо них фирма «Atmel» предлагает использовать новые микроконтроллеры ATmega32x;
- ATmega32, ATmega32L (**Рис. 2.8**) имеют FLASH-память программ объемом 32 Кбайт, ОЗУ объемом 2 Кбайт и EEPROM-память данных объемом 1 Кбайт. Максимальное количество контактов ввода/вывода равно 32. Эти модели полностью (функционально и по цоколевке) совместимы с микроконтроллерами ATmega323(L) и предназначены для замены их в новых разработках;
- ATmega64, ATmega64L (**Рис. 2.9**) имеют FLASH-память программ объемом 64 Кбайт, ОЗУ объемом 4 Кбайт (с возможностью подключения внешнего ОЗУ объемом до 64 Кбайт) и EEPROM-память данных объемом 2 Кбайт. Максимальное количество контактов ввода/вывода равно 53;
- ATmega128, ATmega128L (**Рис. 2.10**) имеют FLASH-память программ объемом 128 Кбайт, ОЗУ объемом 4 Кбайт (с возможностью подключения внешнего ОЗУ объемом до 64 Кбайт) и EEPROM-память данных объемом 4 Кбайт. Максимальное количество контактов ввода/вывода равно 53.

Пока книга готовилась к печати, фирма "Atmel" выпустила еще несколько моделей микроконтроллеров из рассматриваемого семейства — ATmega8535/8535L и ATmega169/169L/169V. Первые две модели имеют FLASH-память программ объемом 8 Кбайт, ОЗУ объемом 512 байт и EEPROM-память данных объемом 512 байт. Количество контактов ввода/вывода в этих моделях равно 32. Микроконтроллеры ATmega169/169L/169V имеют FLASH-память программ объемом 16 Кбайт, ОЗУ объемом 1 Кбайт и EEPROM-память данных объемом 512 байт.

Количество контактов ввода/вывода в этих моделях равно 53. Основным отличием этих моделей от остальных микроконтроллеров семейства Mega является наличие контроллера для управления 4×25 сегментным ЖК-индикатором. В данной книге указанные модели по понятным причинам не описываются, однако основные их параметры приведены в Приложении 2.

Основные параметры всех микроконтроллеров семейства такие, как объем памяти (программ и данных), количество контактов ввода/вывода, тип корпуса, диапазон рабочих частот и напряжение питания, приведены в **Табл. 2.1**. Полная информация по каждой модели приведена в Приложении 1. Дополнительно следует отметить, что все микроконтроллеры семейства Mega выпускаются в коммерческом и промышленном исполнениях. Диапазон рабочих температур при этом 0...+70°C и –40...+85°C соответственно.

Таблица 2.1. **Основные параметры микроконтроллеров AVR семейства Mega**

Обозначение	Память программ (FLASH) [Кбайт]	Память данных (EEPROM) [байт]	Память данных (ОЗУ) [байт]	Количество линий ввода/вывода	Напряжение питания [В]	Тактовая частота [МГц]	Тип корпуса
ATmega8	8	512	1K	23	4.5...5.5	0...16	DIP-28 TQFP-32 MLF-32
ATmega8L					2.7...5.5	0...8	
ATmega8515	8	512	512	35	4.5...5.5	0...16	DIP-40 TQFP-44 PLCC-44 MLF-44
ATmega8515L					2.7...5.5	0...8	
ATmega16	16	512	1K	32	4.5...5.5	0...16	DIP-40 TQFP-44
ATmega16L					2.7...5.5	0...8	
ATmega161	16	512	1K	35	4.0...5.5	0...8	DIP-40 TQFP-44
ATmega161L					2.7...5.5	0...4	
ATmega162	16	512	1K	35	4.5...5.5	0...16	DIP-40 TQFP-44 MLF-44
ATmega162L					2.7...5.5	0...8	
ATmega162V					1.8...3.6	0...1	
ATmega163	16	512	1K	32	4.0...5.5	0...8	DIP-40 TQFP-44
ATmega163L					2.7...5.5	0...4	
ATmega 323	32	1K	2K	32	4.0...5.5	0...8	DIP-40 TQFP-44
ATmega 323L					2.7...5.5	0...4	
ATmega 32	32	1K	2K	32	4.5...5.5	0...16	DIP-40 TQFP-44 MLF-44
ATmega 32L					2.7...5.5	0...8	

Продолжение таблицы 2.1

Обозначение	Память программ (FLASH) [Кбайт]	Память данных (EEPROM) [байт]	Память данных (ОЗУ) [байт]	Количество линий ввода/вывода	Напряжение питания [В]	Тактовая частота [МГц]	Тип корпуса
ATmega64	64	2K	4K	53	4.5...5.5	0...16	TQFP-64
ATmega64L					2.7...5.5	0...8	
ATmega128	128	4K	4K	53	4.5...5.5	0...16	TQFP-64
ATmega128L					2.7...5.5	0...8	

В **Табл. 2.2...2.11** для каждой группы микроконтроллеров приведены названия выводов и указаны их функции (как основные так и дополнительные). 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— выводы питания.

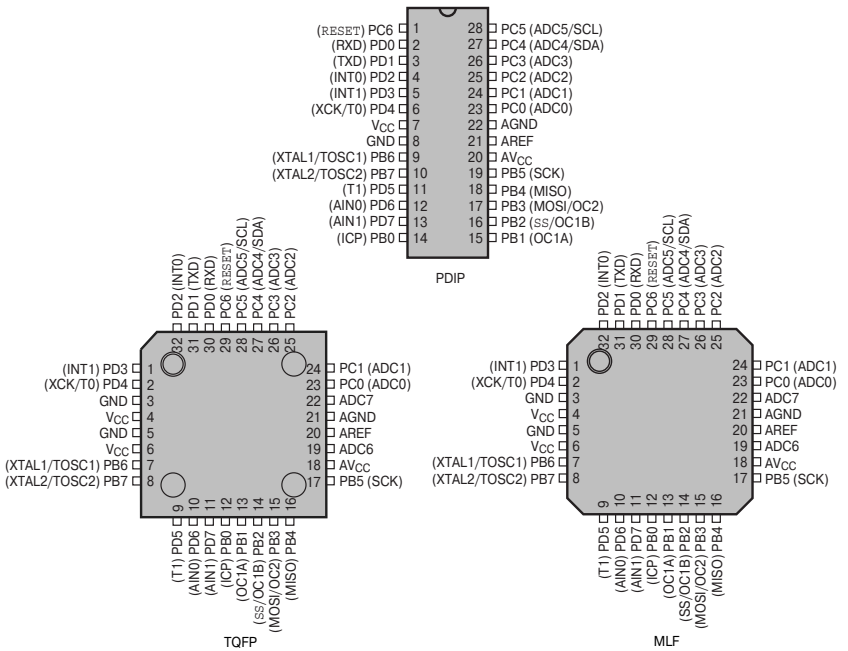


Рис. 2.1. Расположение выводов (вид сверху) модели ATmega8(L)

Таблица 2.2. **Описание выводов модели ATmega8(L)**

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP MLF		
Порт B. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PB0 (ICP)	14	12	I/O	B0 (Вход захвата таймера/счетчика T1 (режим Capture))
PB1 (OC1A)	15	13	I/O	B1 (Выход A таймера/счетчика T1 (режимы Compare, PWM))
PB2 ($\overline{SS}$/OC1B)	16	14	I/O	B2 (Выбор Slave-устройства в канале SPI/ Выход B таймера/счетчика T1 (режимы Compare, PWM))
PB3 (MOSI/OC2)	17	15	I/O	B3 (Выход (Master) или вход (Slave) данных канала SPI/Выход таймера/счетчика T2 (режимы Compare, PWM))
PB4 (MISO)	18	16	I/O	B4 (Вход (Master) или выход (Slave) данных канала SPI)
PB5 (SCK)	19	17	I/O	B5 (Выход (Master) или вход (Slave) тактового сигнала SPI)
PB6 (XTAL1/TOSC1)	9	7	I/O	B6 (Вход тактового генератора/Вывод для подключения резонатора к таймеру/счетчику T2)
PB7 (XTAL2/TOSC2)	10	8	I/O	B7 (Выход тактового генератора/Вывод для подключения резонатора к таймеру/счетчику T2)
Порт C. 7-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PC0 (ADC0)	23	23	I/O	C0 (Вход АЦП)
PC1 (ADC1)	24	24	I/O	C1(Вход АЦП)
PC2 (ADC2)	25	25	I/O	C2 (Вход АЦП)
PC3 (ADC3)	26	26	I/O	C3 (Вход АЦП)
PC4 (ADC4/SDA)	27	27	I/O	C4 (Вход АЦП/Линия данных модуля TWI)

Продолжение таблицы 2.2

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP MLF		
PC5 (ADC5/SCL)	28	28	I/O	C5 (Вход АЦП/Тактовый сигнал модуля TWI)
PC6 ($\overline{\text{RESET}}$)	1	29	I/O	C6 (Вход сброса)
ADC6	—	19	I	Вход АЦП
ADC7	—	22	I	Вход АЦП
Порт D. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PD0 (RXD)	2	30	I/O	D0 (Вход USART)
PD1 (TXD)	3	31	I/O	D1 (Выход USART)
PD2 (INT0)	4	32	I/O	D2 (Вход внешнего прерывания)
PD3 (INT1)	5	1	I/O	D3 (Вход внешнего прерывания)
PD4 (T0/XCK)	6	2	I/O	D4 (Вход внешнего тактового сигнала таймера/счетчика T0/Вход/выход внешнего тактового сигнала USART)
PD5 (T1)	11	9	I/O	D5 (Вход внешнего тактового сигнала таймера/счетчика T1)
PD6 (AIN0)	12	10	I/O	D6 (Положительный вход компаратора)
PD7 (AIN1)	13	11	I/O	D7 (Отрицательный вход компаратора)
AREF	21	20	P	Вход опорного напряжения для АЦП
AGND	22	21	P	Аналоговый общий вывод
AV_{CC}	20	18	P	Вывод источника питания АЦП
GND	8	3, 5	P	Общий вывод
V_{CC}	7	4, 6	P	Вывод источника питания

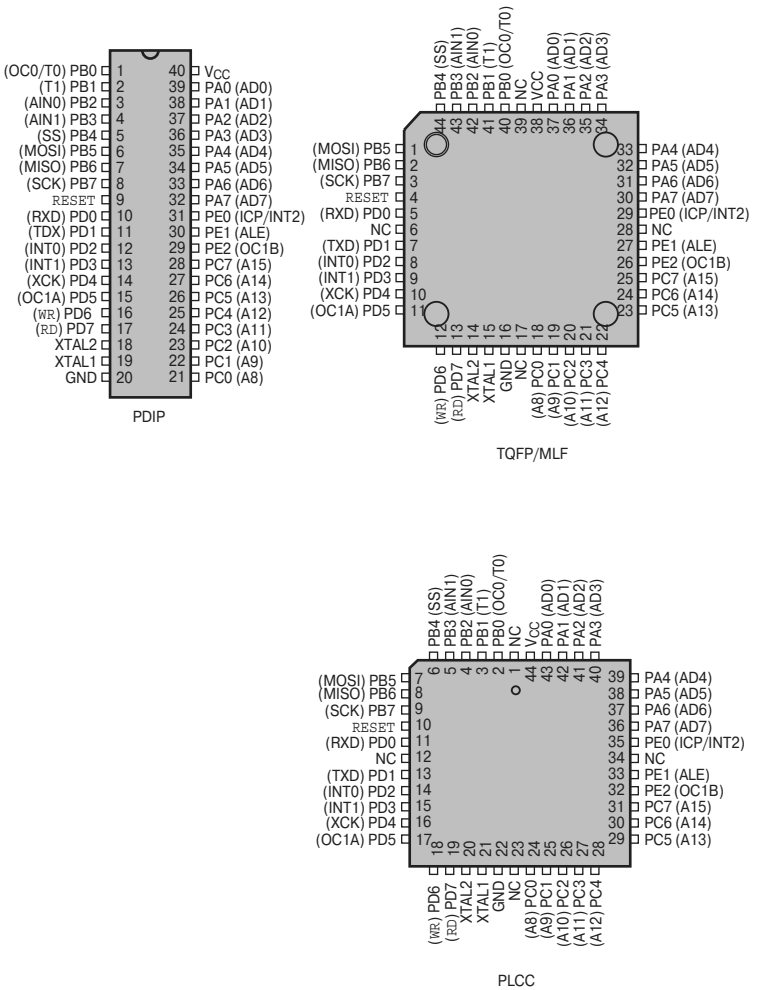


Рис. 2.2. Расположение выводов (вид сверху) моделей ATmega8515(L)

Таблица 2.3. **Описание выводов модели ATmega8515(L)**

Обозначение	Номер вывода			Тип вывода	Описание
	DIP	TQFP MLF	PLCC		
XTAL1	19	15	21	I	Вход тактового генератора
XTAL2	18	14	20	O	Выход тактового генератора
$\overline{\text{RESET}}$	9	4	10	I	Вход сброса. При удержании на входе НИЗКОГО уровня в течение 50 нс выполняется сброс устройства
Порт A. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами					
PA0 (AD0)	39	37	43	I/O	A0 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA1 (AD1)	38	36	42	I/O	A1 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA2 (AD2)	37	35	41	I/O	A2 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA3 (AD3)	36	34	40	I/O	A3 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA4 (AD4)	35	33	39	I/O	A4 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA5 (AD5)	34	32	38	I/O	A5 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA6 (AD6)	33	31	37	I/0	A6 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA7 (AD7)	32	30	36	I/O	A7 (Мультиплексированная ША/ШД для внешнего ОЗУ)
Порт B. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами					
PB0 (T0/OC0)	1	40	2	I/O	B0 (Вход внешнего тактового сигнала таймера/счетчика T0/Выход таймера/счетчика T0 (режимы Compare, PWM))
PB1 (T1)	2	41	3	I/O	B1 (Вход внешнего тактового сигнала таймера/счетчика T1)
PB2 (AIN0)	3	42	4	I/O	B2 (Положительный вход компаратора)
PB3 (AIN1)	4	43	5	I/O	B3 (Отрицательный вход компаратора)
PB4 ($\overline{\text{SS}}$)	5	44	6	I/O	B4 (Выбор Slave-устройства на шине SPI)
PB5 (MOSI)	6	1	7	I/O	B5 (Выход (Master) или вход (Slave) данных модуля SPI)
PB6(MISO)	7	2	8	I/O	B6 (Вход (Master) или выход (Slave) данных модуля SPI)
PB7 (SCK)	8	3	9	I/O	B7 (Выход (Master) или вход (Slave) тактового сигнала модуля SPI)

Продолжение таблицы 2.3

Обозначение	Номер вывода			Тип вывода	Описание
	DIP	TQFP MLF	PLCC		
Порт C. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами					
PC0 (A8)	21	18	24	I/O	C0 (ША для внешнего ОЗУ)
PC1 (A9)	22	19	25	I/O	C1 (ША для внешнего ОЗУ)
PC2 (A10)	23	20	26	I/O	C2 (ША для внешнего ОЗУ)
PC3 (A11)	24	21	27	I/O	C3 (ША для внешнего ОЗУ)
PC4 (A12)	25	22	28	I/O	C4 (ША для внешнего ОЗУ)
PC5 (A13)	26	23	29	I/O	C5 (ША для внешнего ОЗУ)
PC6 (A14)	27	24	30	I/O	C6 (ША для внешнего ОЗУ)
PC7 (A15)	28	25	31	I/O	C7 (ША для внешнего ОЗУ)
Порт D. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами					
PD0 (RXD)	10	5	11	I/O	D0 (Вход USART)
PD1 (TXD)	11	7	13	I/O	D1 (Выход USART)
PD2 (INT0)	12	8	14	I/O	D2 (Вход внешнего прерывания)
PD3 (INT1)	13	9	15	I/O	D3 (Вход внешнего прерывания)
PD4 (XCK)	14	10	16	I/O	D4 (Вход/выход внешнего тактового сигнала USART)
PD5 (OC1A)	15	11	17	I/O	D5 (Выход A таймера/счетчика T1 (режимы Compare, PWM))
PD6 ($\overline{WR}$)	16	12	18	I/O	D6 (Строб записи во внешнее ОЗУ)
PD7 ($\overline{RD}$)	17	13	19	I/O	D7 (Строб чтения из внешнего ОЗУ)
Порт E. 3-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами					
PE0 (ICP/INT2)	31	29	35	I/O	E0 (Вход захвата таймера/счетчика T1 (режим Capture)/Вход внешнего прерывания)
PE1 (ALE)	30	27	33	I/O	E1 (Строб адреса внешнего ОЗУ)
PE2 (OC1B)	29	26	32	I/O	E2 (Выход B таймера/счетчика T1 (режимы Compare, PWM))
GND	20	16	22	P	Общий вывод
V_{CC}	40	38	44	P	Вывод источника питания
NC	—	6, 17, 28, 39	1, 12, 23, 34	—	Не используются

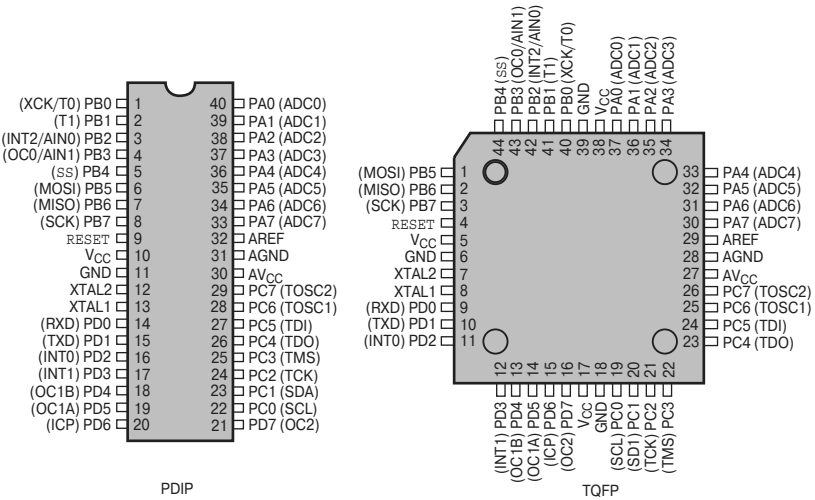


Рис. 2.3. Расположение выводов (вид сверху) моделей ATmega16(L)

Таблица 2.4. **Описание выводов модели ATmega16(L)**

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP		
XTAL1	13	8	I	Вход тактового генератора
XTAL2	12	7	O	Выход тактового генератора
$\overline{\text{RESET}}$	9	4	I	Вход сброса. При удержании на входе НИЗКОГО уровня в течение 50 нс выполняется сброс устройства
Порт A. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PA0 (ADC0)	40	37	I/O	A0 (Вход АЦП)
PA1 (ADC1)	39	36	I/O	A1 (Вход АЦП)
PA2 (ADC2)	38	35	I/O	A2 (Вход АЦП)
PA3 (ADC3)	37	34	I/O	A3 (Вход АЦП)
PA4 (ADC4)	36	33	I/O	A4 (Вход АЦП)
PA5 (ADC5)	35	32	I/O	A5 (Вход АЦП)
PA6 (ADC6)	34	31	I/0	A6 (Вход АЦП)
PA7 (ADC7)	33	30	I/O	A7 (Вход АЦП)
Порт B. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PB0 (T0/XCK)	1	40	I/O	B0 (Вход внешнего тактового сигнала таймера/счетчика T0/Вход/выход внешнего тактового сигнала USART)

Продолжение таблицы 2.4

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP		
PB1 (T1)	2	41	I/O	B1 (Вход внешнего тактового сигнала таймера/счетчика T1)
PB2 (AIN0/INT2)	3	42	I/O	B2 (Положительный вход компаратора/Вход внешнего прерывания)
PB3 (AIN1/OC0)	4	43	I/O	B3 (Отрицательный вход компаратора/Выход таймера/счетчика T0 (режимы Compare, PWM))
PB4 ($\overline{SS}$)	5	44	I/O	B4 (Выбор Slave-устройства на шине SPI)
PB5 (MOSI)	6	1	I/O	B5 (Выход (Master) или вход (Slave) данных модуля SPI)
PB6(MISO)	7	2	I/O	B6 (Вход (Master) или выход (Slave) данных модуля SPI)
PB7 (SCK)	8	3	I/O	B7 (Выход (Master) или вход (Slave) тактового сигнала модуля SPI)
Порт C. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PC0 (SCL)	22	19	I/O	C0 (Тактовый сигнал модуля TWI)
PC1 (SDA)	23	20	I/O	C1 (Линия данных модуля TWI)
PC2 (TCK)	24	21	I/O	C2 (Тактовый сигнал JTAG)
PC3 (TMS)	25	22	I/O	C3 (Выбор режима JTAG)
PC4 (TDO)	26	23	I/O	C4 (Выход данных JTAG)
PC5 (TDI)	27	24	I/O	C5 (Вход данных JTAG)
PC6 (TOSC1)	28	25	I/O	C6 (Выход для подключения резонатора к таймеру/счетчику T2)
PC7 (TOSC2)	29	26	I/O	C7 (Вход для подключения резонатора к таймеру)/счетчику T2)
Порт D. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PD0 (RXD)	14	9	I/O	D0 (Вход USART)
PD1 (TXD)	15	10	I/O	D1 (Выход USART)
PD2 (INT0)	16	11	I/O	D2 (Вход внешнего прерывания)
PD3 (INT1)	17	12	I/O	D3 (Вход внешнего прерывания)
PD4 (OC1B)	18	13	I/O	D4 (Выход B таймера/счетчика T1 (режимы Compare, PWM))
PD5 (OC1A)	19	14	I/O	D5 (Выход A таймера/счетчика T1 (режимы Compare, PWM))
PD6 (ICP)	20	15	I/O	D6 (Вход захвата таймера/счетчика T1 (режим Capture))
PD7 (OC2)	21	16	I/O	D7 (Выход таймера/счетчика T2 (режимы Compare, PWM))
AREF	32	29	P	Вход опорного напряжения для АЦП
AGND	31	28	P	Общий вывод (аналоговый)
AV_{CC}	30	27	P	Вывод источника питания АЦП
GND	11	6, 18, 39	P	Общий вывод
V_{CC}	10	5, 17, 38	P	Вывод источника питания

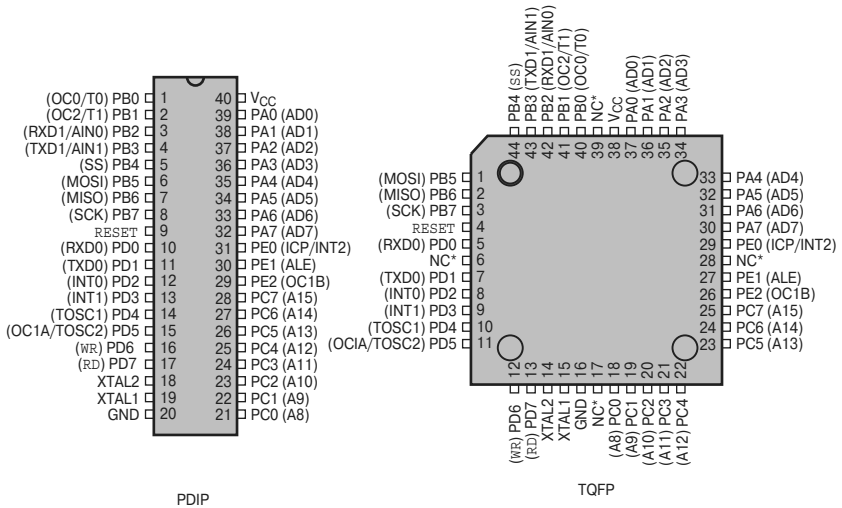


Рис. 2.4. Расположение выводов (вид сверху) моделей ATmega161(L)

Таблица 2.5. **Описание выводов модели ATmega161(L)**

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP		
XTAL1	19	15	I	Вход тактового генератора
XTAL2	18	14	O	Выход тактового генератора
$\overline{\text{RESET}}$	9	4	I	Вход сброса. При удержании на входе НИЗКОГО уровня в течение 50 нс выполняется сброс устройства
Порт A. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PA0 (AD0)	39	37	I/O	A0 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA1 (AD1)	38	36	I/O	A1 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA2 (AD2)	37	35	I/O	A2 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA3 (AD3)	36	34	I/O	A3 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA4 (AD4)	35	33	I/O	A4 (Мультиплексированная ША/ШД для внешнего ОЗУ)

Продолжение таблицы 2.5

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP		
PA5 (AD5)	34	32	I/O	A5 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA6 (AD6)	33	31	I/0	A6 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA7 (AD7)	32	30	I/O	A7 (Мультиплексированная ША/ШД для внешнего ОЗУ)
Порт B. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PB0 (OC0/T0)	1	40	I/O	B0 (Выход таймера/счетчика T0 (режимы Compare, PWM)/Вход внешнего тактового сигнала таймера/счетчика T0)
PB1 (OC2/T1)	2	41	I/O	B1 (Выход таймера/счетчика T2 (режимы Compare, PWM)/Вход внешнего тактового сигнала таймера/счетчика T1)
PB2 (RXD1/AIN0)	3	42	I/O	B2 (Вход 2-го UART/Положительный вход компаратора)
PB3 (TXD1/AIN1)	4	43	I/O	B3 (Выход 2-го UART/Отрицательный вход компаратора)
PB4 ($\overline{SS}$)	5	44	I/O	B4 (Выбор Slave-устройства на шине SPI)
PB5 (MOSI)	6	1	I/O	B5 (Выход (Master) или вход (Slave) данных модуля SPI)
PB6(MISO)	7	2	I/O	B6 (Вход (Master) или выход (Slave) данных модуля SPI)
PB7 (SCK)	8	3	I/O	B7 (Выход (Master) или вход (Slave) тактового сигнала модуля SPI)
Порт C. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PC0 (A8)	21	18	I/O	C0 (ША для внешнего ОЗУ)
PC1 (A9)	22	19	I/O	C1 (ША для внешнего ОЗУ)
PC2 (A10)	23	20	I/O	C2 (ША для внешнего ОЗУ)
PC3 (A11)	24	21	I/O	C3 (ША для внешнего ОЗУ)
PC4 (A12)	25	22	I/O	C4 (ША для внешнего ОЗУ)

Продолжение таблицы 2.5

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP		
PC5 (A13)	26	23	I/O	C5 (ША для внешнего ОЗУ)
PC6 (A14)	27	24	I/O	C6 (ША для внешнего ОЗУ)
PC7 (A15)	28	25	I/O	C7 (ША для внешнего ОЗУ)
Порт D. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PD0 (RXD0)	10	5	I/O	D0 (Вход 1-го UART)
PD1 (TXD0)	11	7	I/O	D1 (Выход 1-го UART)
PD2 (INT0)	12	8	I/O	D2 (Вход внешнего прерывания)
PD3 (INT1)	13	9	I/O	D3 (Вход внешнего прерывания)
PD4 (TOSC1)	14	10	I/O	D4 (Вход для подключения резонатора к таймеру/счетчику T2)
PD5 (TOSC2/OC1A)	15	11	I/O	D5 (Выход для подключения резонатора к таймеру/счетчику T2/Выход A таймера/счетчика T1 (режимы Compare, PWM))
PD6 ($\overline{WR}$)	16	12	I/O	D6 (Строб записи во внешнее ОЗУ)
PD7 ($\overline{RD}$)	17	13	I/O	D7 (Строб чтения из внешнего ОЗУ)
Порт E. 3-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PE0 (ICP/INT2)	31	29	I/O	E0 (Вход захвата таймера/счетчика T1 (режим Capture)/Вход внешнего прерывания)
PE1 (ALE)	30	27	I/O	E1 (Строб адреса внешнего ОЗУ)
PE2 (OC1B)	29	26	I/O	E2 (Выход B таймера/счетчика T1 (режимы Compare, PWM))
GND	20	16	P	Общий вывод
V_{CC}	40	38	P	Вывод источника питания
NC	—	6, 17, 28, 39	—	Не используются

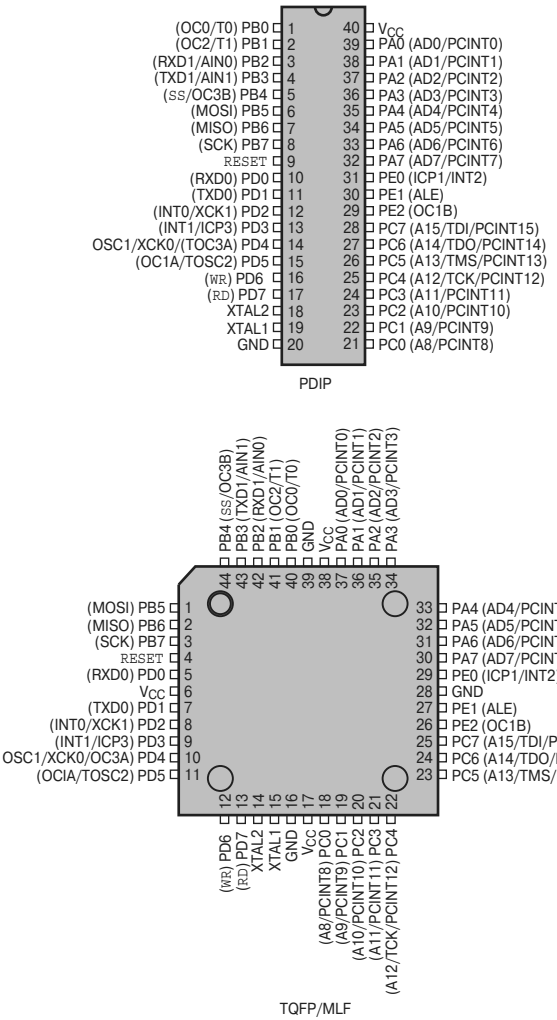


Рис. 2.5. Расположение выводов (вид сверху) моделей ATmega162(L, V)

Таблица 2.6. **Описание выводов модели ATmega162(L, V)**

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP MLF		
XTAL1	19	15	I	Вход тактового генератора
XTAL2	18	14	O	Выход тактового генератора
$\overline{\text{RESET}}$	9	4	I	Вход сброса. При удержании на входе НИЗКОГО уровня в течение 50 нс выполняется сброс устройства
Порт A. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PA0 (AD0/PCINT0)	39	37	I/O	A0 (Мультиплексированная ША/ШД для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала)
PA1 (AD1/PCINT1)	38	36	I/O	A1 (Мультиплексированная ША/ШД для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала)
PA2 (AD2/PCINT2)	37	35	I/O	A2 (Мультиплексированная ША/ШД для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала)
PA3 (AD3/PCINT3)	36	34	I/O	A3 (Мультиплексированная ША/ШД для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала)
PA4 (AD4/PCINT4)	35	33	I/O	A4 (Мультиплексированная ША/ШД для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала)
PA5 (AD5/PCINT5)	34	32	I/O	A5 (Мультиплексированная ША/ШД для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала)
PA6 (AD6/PCINT6)	33	31	I/0	A6 (Мультиплексированная ША/ШД для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала)
PA7 (AD7/PCINT7)	32	30	I/O	A7 (Мультиплексированная ША/ШД для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала)
Порт B. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PB0 (OC0/T0)	1	40	I/O	B0 (Выход таймера/счетчика T0 (режимы Compare, PWM)/Вход внешнего тактового сигнала таймера/счетчика T0)

Продолжение таблицы 2.6

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP MLF		
PB1 (OC2/T1)	2	41	I/O	B1 (Выход таймера/счетчика T2 (режимы Compare, PWM)/Вход внешнего тактового сигнала таймера/счетчика T1)
PB2 (RXD1/AIN0)	3	42	I/O	B2 (Вход 2-го USART/Положительный вход компаратора)
PB3 (TXD1/AIN1)	4	43	I/O	B3 (Выход 2-го USART/Отрицательный вход компаратора)
PB4 ($\overline{SS}$)	5	44	I/O	B4 (Выбор Slave-устройства на шине SPI/Выход таймера/счетчика T3 (режимы Compare, PWM))
PB5 (MOSI)	6	1	I/O	B5 (Выход (Master)/Вход (Slave) данных модуля SPI)
PB6(MISO)	7	2	I/O	B6 (Вход (Master)/Выход (Slave) данных модуля SPI)
PB7 (SCK)	8	3	I/O	B7 (Выход (Master)/Вход (Slave) тактового сигнала модуля SPI)
Порт C. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PC0 (A8/PCINT8)	21	18	I/O	C0 (ША для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала)
PC1 (A9/PCINT9)	22	19	I/O	C1 (ША для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала)
PC2 (A10/PCINT10)	23	20	I/O	C2 (ША для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала)
PC3 (A11/PCINT11)	24	21	I/O	C3 (ША для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала)
PC4 (A12/PCINT2/TCK)	25	22	I/O	C4 (ША для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала или тактовый сигнал JTAG)
PC5 (A13/PCINT13/TMS)	26	23	I/O	C5 (ША для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала или выбор режима JTAG)
PC6 (A14/PCINT14/TDO)	27	24	I/O	C6 (ША для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала или выход данных JTAG)

Продолжение таблицы 2.6

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP MLF		
PC7 (A15/PCINT15/TDI)	28	25	I/O	C7 (ША для внешнего ОЗУ/Вход внешнего прерывания по изменению сигнала или вход данных JTAG)
Порт D. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PD0 (RXD0)	10	5	I/O	D0 (Вход 1-го USART)
PD1 (TXD0)	11	7	I/O	D1 (Выход 1-го USART)
PD2 (INT0/XCK1)	12	8	I/O	D2 (Вход внешнего прерывания/Вход/выход внешнего тактового сигнала 2-го USART)
PD3 (INT1/ICP3)	13	9	I/O	D3 (Вход внешнего прерывания/Вход захвата таймера/счетчика T3 (режим Capture))
PD4 (TOSC1/XCK0/OC3A)	14	10	I/O	D4 (Вход для подключения резонатора к таймеру/счетчику T2/Вход/выход внешнего тактового сигнала 1-го USART или выход A таймера/счетчика T3 (режимы Compare, PWM))
PD5 (TOSC2/OC1A)	15	11	I/O	D5 (Вход для подключения резонатора к таймеру/счетчику T2/Выход A таймера/счетчика T1 (режимы Compare, PWM))
PD6 ($\overline{WR}$)	16	12	I/O	D6 (Строб записи во внешнее ОЗУ)
PD7 ($\overline{RD}$)	17	13	I/O	D7 (Строб чтения из внешнего ОЗУ)
Порт E. 3-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PE0 (ICP1/INT2)	31	29	I/O	E0 (Вход захвата таймера/счетчика T1 (режим Capture)/Вход внешнего прерывания)
PE1 (ALE)	30	27	I/O	E1 (Строб адреса внешнего ОЗУ)
PE2 (OC1B)	29	26	I/O	E2 (Выход B таймера/счетчика T1 (режимы Compare, PWM))
GND	20	16, 28, 39	P	Общий вывод
V_{CC}	40	6, 17, 38	P	Вывод источника питания

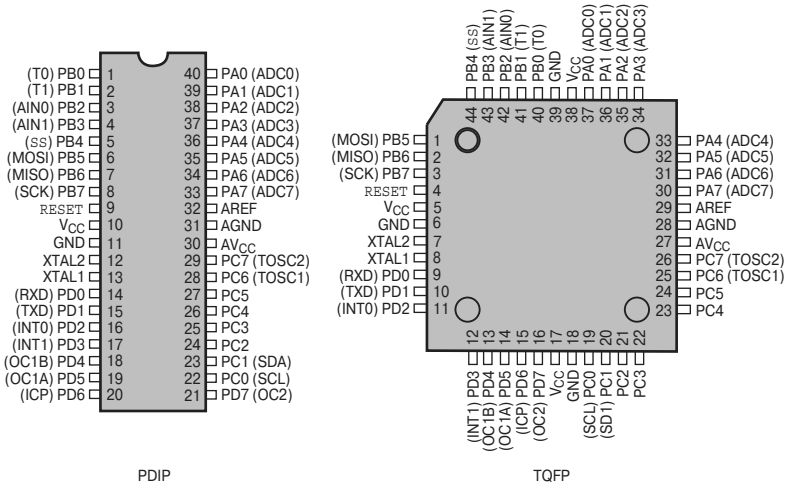


Рис. 2.6. Расположение выводов (вид сверху) моделей ATmega163(L)

Таблица 2.7. **Описание выводов модели ATmega163(L)**

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP		
XTAL1	13	8	I	Вход тактового генератора
XTAL2	12	7	O	Выход тактового генератора
$\overline{\text{RESET}}$	9	4	I	Вход сброса. При удержании на входе НИЗКОГО уровня в течение 50 нс выполняется сброс устройства
Порт A. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PA0 (ADC0)	40	37	I/O	A0 (Вход АЦП)
PA1 (ADC1)	39	36	I/O	A1 (Вход АЦП)
PA2 (ADC2)	38	35	I/O	A2 (Вход АЦП)
PA3 (ADC3)	37	34	I/O	A3 (Вход АЦП)
PA4 (ADC4)	36	33	I/O	A4 (Вход АЦП)
PA5 (ADC5)	35	32	I/O	A5 (Вход АЦП)
PA6 (ADC6)	34	31	I/0	A6 (Вход АЦП)
PA7 (ADC7)	33	30	I/O	A7 (Вход АЦП)
Порт B. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PB0 (T0)	1	40	I/O	B0 (Вход внешнего тактового сигнала таймера/счетчика T0)
PB1 (T1)	2	41	I/O	B1 (Вход внешнего тактового сигнала таймера/счетчика T1)
PB2 (AIN0)	3	42	I/O	B2 (Положительный вход компаратора)

Продолжение таблицы 2.7

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP		
PB3 (AIN1)	4	43	I/O	B3 (Отрицательный вход компаратора)
PB4 ($\overline{SS}$)	5	44	I/O	B4 (Выбор Slave-устройства на шине SPI)
PB5 (MOSI)	6	1	I/O	B5 (Выход (Master) или вход (Slave) данных модуля SPI)
PB6(MISO)	7	2	I/O	B6 (Вход (Master) или выход (Slave) данных модуля SPI)
PB7 (SCK)	8	3	I/O	B7 (Выход (Master) или вход (Slave) тактового сигнала модуля SPI)
Порт C. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PC0 (SCL)	22	19	I/O	C0 (Тактовый сигнал модуля TWI)
PC1 (SDA)	23	20	I/O	C1 (Линия данных модуля TWI)
PC2	24	21	I/O	C2
PC3	25	22	I/O	C3
PC4	26	23	I/O	C4
PC5	27	24	I/O	C5
PC6 (TOSC1)	28	25	I/O	C6 (Выход для подключения резонатора к таймеру/счетчику T2)
PC7 (TOSC2)	29	26	I/O	C7 (Вход для подключения резонатора к таймеру/счетчику T2)
Порт D. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PD0 (RXD)	14	9	I/O	D0 (Вход UART)
PD1 (TXD)	15	10	I/O	D1 (Выход UART)
PD2 (INT0)	16	11	I/O	D2 (Вход внешнего прерывания)
PD3 (INT1)	17	12	I/O	D3 (Вход внешнего прерывания)
PD4 (OC1B)	18	13	I/O	D4 (Выход B таймера/счетчика T1 (режимы Compare, PWM))
PD5 (OC1A)	19	14	I/O	D5 (Выход A таймера/счетчика T1 (режимы Compare, PWM))
PD6 (ICP)	20	15	I/O	D6 (Вход захвата таймера/счетчика T1 (режим Capture))
PD7 (OC2)	21	16	I/O	D7 (Выход таймера/счетчика T2 (режимы Compare, PWM))
AREF	32	29	P	Вход опорного напряжения для АЦП
AGND	31	28	P	Аналоговый общий вывод
AV_{CC}	30	27	P	Вывод источника питания АЦП
GND	11	6, 18, 39	P	Общий вывод
V_{CC}	10	5, 17, 38	P	Вывод источника питания

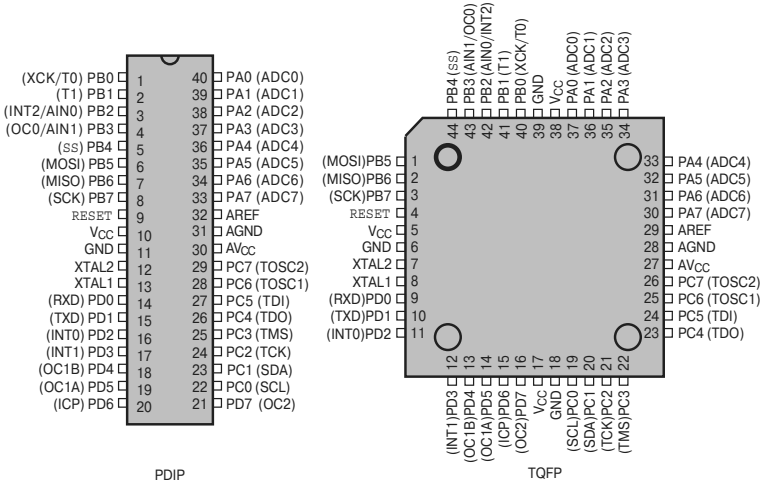


Рис. 2.7. Расположение выводов (вид сверху) моделей ATmega323(L)

Таблица 2.8. **Описание выводов модели ATmega323(L)**

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP		
XTAL1	13	8	I	Вход тактового генератора
XTAL2	12	7	O	Выход тактового генератора
$\overline{\text{RESET}}$	9	4	I	Вход сброса. При удержании на входе НИЗКОГО уровня в течение 50 нс выполняется сброс устройства
Порт A. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PA0 (ADC0)	40	37	I/O	A0 (Вход АЦП)
PA1 (ADC1)	39	36	I/O	A1 (Вход АЦП)
PA2 (ADC2)	38	35	I/O	A2 (Вход АЦП)
PA3 (ADC3)	37	34	I/O	A3 (Вход АЦП)
PA4 (ADC4)	36	33	I/O	A4 (Вход АЦП)
PA5 (ADC5)	35	32	I/O	A5 (Вход АЦП)
PA6 (ADC6)	34	31	I/0	A6 (Вход АЦП)
PA7 (ADC7)	33	30	I/O	A7 (Вход АЦП)
Порт B. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PB0 (T0/XCK)	1	40	I/O	B0 (Вход внешнего тактового сигнала таймера/счетчика T0 или вход/выход внешнего тактового сигнала USART)
PB1 (T1)	2	41	I/O	B1 (Вход внешнего тактового сигнала таймера/счетчика T1)
PB2 (AIN0/INT2)	3	42	I/O	B2 (Положительный вход компаратора/Вход внешнего прерывания)

Продолжение таблицы 2.8

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP		
PB3 (AIN1/OC0)	4	43	I/O	B3 (Отрицательный вход компаратора/Выход таймера/счетчика T0 (режимы Compare, PWM)
PB4 ($\overline{SS}$)	5	44	I/O	B4 (Выбор Slave-устройства на шине SPI)
PB5 (MOSI)	6	1	I/O	B5 (Выход (Master) или вход (Slave) данных модуля SPI)
PB6(MISO)	7	2	I/O	B6 (Вход (Master) или выход (Slave) данных модуля SPI)
PB7 (SCK)	8	3	I/O	B7 (Выход (Master) или вход (Slave) тактового сигнала модуля SPI)
Порт C. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PC0 (SCL)	22	19	I/O	C0 (Тактовый сигнал модуля TWI)
PC1 (SDA)	23	20	I/O	C1 (Линия данных модуля TWI)
PC2 (TCK)	24	21	I/O	C2 (Тактовый сигнал JTAG)
PC3 (TMS)	25	22	I/O	C3 (Выбор режима JTAG)
PC4 (TDO)	26	23	I/O	C4 (Выход данных JTAG)
PC5 (TDI)	27	24	I/O	C5 (Вход данных JTAG)
PC6 (TOSC1)	28	25	I/O	C6 (Выход для подключения резонатора к таймеру/счетчику T2)
PC7 (TOSC2)	29	26	I/O	C7 (Вход для подключения резонатора к таймеру/счетчику T2)
Порт D. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PD0 (RXD)	14	9	I/O	D0 (Вход UART)
PD1 (TXD)	15	10	I/O	D1 (Выход UART)
PD2 (INT0)	16	11	I/O	D2 (Вход внешнего прерывания)
PD3 (INT1)	17	12	I/O	D3 (Вход внешнего прерывания)
PD4 (OC1B)	18	13	I/O	D4 (Выход B таймера/счетчика T1 (режимы Compare, PWM))
PD5 (OC1A)	19	14	I/O	D5 (Выход A таймера/счетчика T1 (режимы Compare, PWM))
PD6 (ICP)	20	15	I/O	D6 (Вход захвата таймера/счетчика T1 (режим Capture))
PD7 (OC2)	21	16	I/O	D7 (Выход таймера/счетчика T2 (режимы Compare, PWM))
AREF	32	29	P	Вход опорного напряжения для АЦП
AGND	31	28	P	Аналоговый общий вывод
AV_{CC}	30	27	P	Вывод источника питания АЦП
GND	11	6, 18, 39	P	Общий вывод
V_{CC}	10	5, 17, 38	P	Вывод источника питания

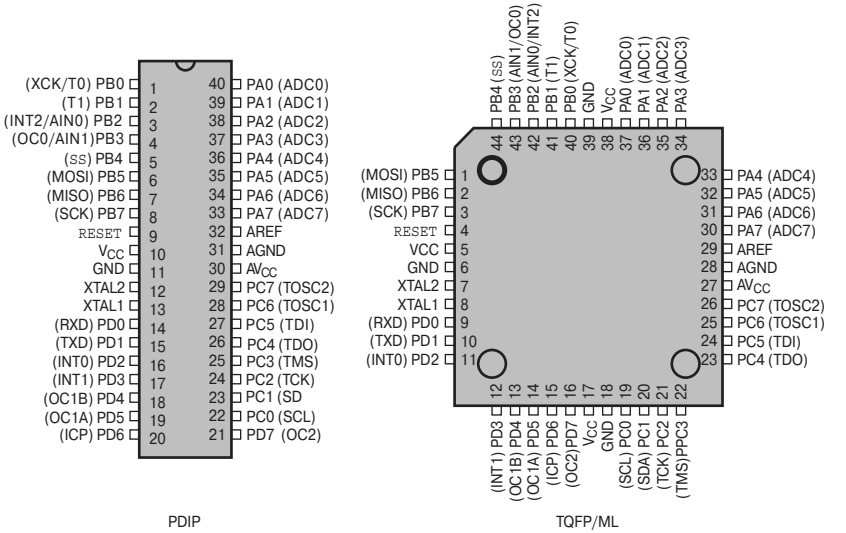


Рис. 2.8. Расположение выводов (вид сверху) моделей ATmega32(L)

Таблица 2.9. **Описание выводов модели ATmega32(L)**

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP		
XTAL1	13	8	I	Вход тактового генератора
XTAL2	12	7	O	Выход тактового генератора
$\overline{\text{RESET}}$	9	4	I	Вход сброса. При удержании на входе НИЗКОГО уровня в течение 50 нс выполняется сброс устройства
Порт A. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PA0 (ADC0)	40	37	I/O	A0 (Вход АЦП)
PA1 (ADC1)	39	36	I/O	A1 (Вход АЦП)
PA2 (ADC2)	38	35	I/O	A2 (Вход АЦП)
PA3 (ADC3)	37	34	I/O	A3 (Вход АЦП)
PA4 (ADC4)	36	33	I/O	A4 (Вход АЦП)
PA5 (ADC5)	35	32	I/O	A5 (Вход АЦП)
PA6 (ADC6)	34	31	I/0	A6 (Вход АЦП)
PA7 (ADC7)	33	30	I/O	A7 (Вход АЦП)
Порт B. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PB0 (T0/XCK)	1	40	I/O	B0 (Вход внешнего тактового сигнала таймера/счетчика T0/Вход/выход внешнего тактового сигнала USART)

Продолжение таблицы 2.9

Обозначение	Номер вывода		Тип вывода	Описание
	DIP	TQFP		
PB1 (T1)	2	41	I/O	B1 (Вход внешнего тактового сигнала таймера/счетчика T1)
PB2 (AIN0/INT2)	3	42	I/O	B2 (Положительный вход компаратора/Вход внешнего прерывания)
PB3 (AIN1/OC0)	4	43	I/O	B3 (Отрицательный вход компаратора/Выход таймера/счетчика T0 (режимы Compare, PWM))
PB4 ($\overline{SS}$)	5	44	I/O	B4 (Выбор Slave-устройства на шине SPI)
PB5 (MOSI)	6	1	I/O	B5 (Выход (Master) или вход (Slave) данных модуля SPI)
PB6(MISO)	7	2	I/O	B6 (Вход (Master) или выход (Slave) данных модуля SPI)
PB7 (SCK)	8	3	I/O	B7 (Выход (Master) или вход (Slave) тактового сигнала модуля SPI)
Порт C. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PC0 (SCL)	22	19	I/O	C0 (Тактовый сигнал модуля TWI)
PC1 (SDA)	23	20	I/O	C1 (Линия данных модуля TWI)
PC2 (TCK)	24	21	I/O	C2 (Тактовый сигнал JTAG)
PC3 (TMS)	25	22	I/O	C3 (Выбор режима JTAG)
PC4 (TDO)	26	23	I/O	C4 (Выход данных JTAG)
PC5 (TDI)	27	24	I/O	C5 (Вход данных JTAG)
PC6 (TOSC1)	28	25	I/O	C6 (Выход для подключения резонатора к таймеру/счетчику T2)
PC7 (TOSC2)	29	26	I/O	C7 (Вход для подключения резонатора к таймеру/счетчику T2)
Порт D. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами				
PD0 (RXD)	14	9	I/O	D0 (Вход UART)
PD1 (TXD)	15	10	I/O	D1 (Выход UART)
PD2 (INT0)	16	11	I/O	D2 (Вход внешнего прерывания)
PD3 (INT1)	17	12	I/O	D3 (Вход внешнего прерывания)
PD4 (OC1B)	18	13	I/O	D4 (Выход B таймера/счетчика T1 (режимы Compare, PWM))
PD5 (OC1A)	19	14	I/O	D5 (Выход A таймера/счетчика T1 (режимы Compare, PWM))
PD6 (ICP)	20	15	I/O	D6 (Вход захвата таймера/счетчика T1 (режим Capture))
PD7 (OC2)	21	16	I/O	D7 (Выход таймера/счетчика T2 (режимы Compare, PWM))
AREF	32	29	P	Вход опорного напряжения для АЦП
AGND	31	28	P	Аналоговый общий вывод
AV_{CC}	30	27	P	Вывод источника питания АЦП
GND	11	6, 18, 39	P	Общий вывод
V_{CC}	10	5, 17, 38	P	Вывод источника питания

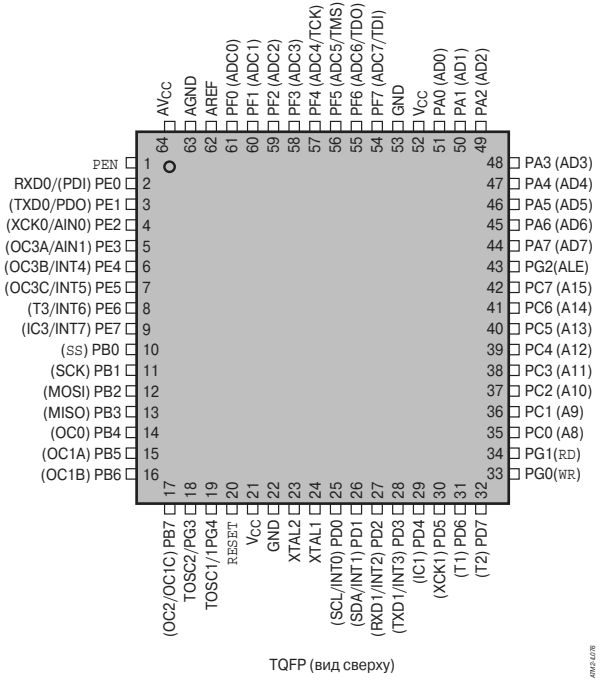


Рис. 2.9. Расположение выводов модели ATmega64(L)

Таблица 2.10. **Описание выводов модели ATmega64(L)**

Обозначение	Номер вывода	Тип вывода	Описание
XTAL1	24	I	Вход тактового генератора
XTAL2	23	O	Выход тактового генератора
$\overline{\text{RESET}}$	20	I	Вход сброса. При удержании на входе НИЗКОГО уровня в течение 50 нс выполняется сброс устройства
Порт А. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PA0 (AD0)	51	I/O	A0 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA1 (AD1)	50	I/O	A1 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA2 (AD2)	49	I/O	A2 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA3 (AD3)	48	I/O	A3 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA4 (AD4)	47	I/O	A4 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA5 (AD5)	46	I/O	A5 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA6 (AD6)	45	I/0	A6 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA7 (AD7)	44	I/O	A7 (Мультиплексированная ША/ШД для внешнего ОЗУ)

Продолжение таблицы 2.10

Обозначение	Номер вывода	Тип вывода	Описание
Порт B. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PB0 ($\overline{SS}$)	10	I/O	B0 (Выбор Slave-устройства на шине SPI)
PB1 (SCK)	11	I/O	B1 (Выход (Master) или вход (Slave) тактового сигнала модуля SPI)
PB2 (MOSI)	12	I/O	B2 (Выход (Master) или вход (Slave) данных модуля SPI)
PB3 (MISO)	13	I/O	B3 (Вход (Master) или выход (Slave) данных модуля SPI)
PB4 (OC0)	14	I/O	B4 (Выход таймера/счетчика T0 (режимы Compare, PWM))
PB5 (OC1A)	15	I/O	B5 (Выход A таймера/счетчика T1 (режимы Compare, PWM))
PB6 (OC1B)	16	I/O	B6 (Выход B таймера/счетчика T1 (режимы Compare, PWM))
PB7 (OC2/OC1C)	17	I/O	B7 (Выход таймера/счетчика T2 (режимы Compare, PWM)/Выход C таймера/счетчика T1 (режимы Compare, PWM))
Порт C. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PC0 (A8)	35	I/O	C0 (ША для внешнего ОЗУ)
PC1 (A9)	36	I/O	C1 (ША для внешнего ОЗУ)
PC2 (A10)	37	I/O	C2 (ША для внешнего ОЗУ)
PC3 (A11)	38	I/O	C3 (ША для внешнего ОЗУ)
PC4 (A12)	39	I/O	C4 (ША для внешнего ОЗУ)
PC5 (A13)	40	I/O	C5 (ША для внешнего ОЗУ)
PC6 (A14)	41	I/O	C6 (ША для внешнего ОЗУ)
PC7 (A15)	42	I/O	C7 (ША для внешнего ОЗУ)
Порт D. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PD0 (SCL/INT0)	25	I/O	D0 (Тактовый сигнал модуля TWI/Вход внешнего прерывания)
PD1 (SDA/INT1)	26	I/O	D1 (Линия данных модуля TWI/Вход внешнего прерывания)
PD2 (RXD1/INT2)	27	I/O	D2 (Вход 2-го USART)/Вход внешнего прерывания)
PD3 (TXD1/INT3)	28	I/O	D3 (Выход 2-го USART/Вход внешнего прерывания)
PD4 (IC1)	29	I/O	D4 (Вход захвата таймера/счетчика T1 (режим Capture))
PD5 (XCK1)	30	I/O	D5 (Вход/выход внешнего тактового сигнала 2-го USART)
PD6 (T1)	31	I/O	D6 (Вход внешнего тактового сигнала таймера/счетчика T1)
PD7 (T2)	32	I/O	D7 (Вход внешнего тактового сигнала таймера/счетчика T2)

Продолжение таблицы 2.10

Обозначение	Номер вывода	Тип вывода	Описание
Порт E. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PE0 (RXD0/PDI)	2	I/O	E0 (Вход 1-го USART/Вход данных при последовательном программировании)
PE1 (TXD0/PDO)	3	I/O	E1 (Выход 1-го USART/Выход данных при последовательном программировании)
PE2 (XCK0/AIN0)	4	I/O	E2 (Вход/выход внешнего тактового сигнала 1-го USART/Положительный вход компаратора)
PE3 (OC3A/AIN1)	5	I/O	E3 (Выход A таймера/счетчика T3 (режимы Compare, PWM)/Отрицательный вход компаратора)
PE4 (OC3B/INT4)	6	I/O	E4 (Выход B таймера/счетчика T3 (режимы Compare, PWM)/Вход внешнего прерывания)
PE5 (OC3C/INT5)	7	I/O	E5 (Выход C таймера/счетчика T3 (режимы Compare, PWM)/Вход внешнего прерывания)
PE6 (T3/INT6)	8	I/O	E6 (Вход внешнего тактового сигнала таймера/счетчика T1/Вход внешнего прерывания)
PE7 (IC3/INT7)	9	I/O	E7 (Вход захвата таймера/счетчика T1 (режим Capture)/Вход внешнего прерывания)
Порт F. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PF0 (ADC0)	61	I/O	F0 (Вход АЦП)
PF1 (ADC1)	60	I/O	F1 (Вход АЦП)
PF2 (ADC2)	59	I/O	F2 (Вход АЦП)
PF3 (ADC3)	58	I/O	F3 (Вход АЦП)
PF4 (ADC4/TCK)	57	I/O	F4 (Вход АЦП/Тактовый сигнал JTAG)
PF5 (ADC5/TMS)	56	I/O	F5 (Вход АЦП/Выбор режима JTAG)
PF6 (ADC6/TDO)	55	I/0	F6 (Вход АЦП/Выход данных JTAG)
PF7 (ADC7/TDI)	54	I/O	F7 (Вход АЦП/Вход данных JTAG)
Порт G. 5-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PG0($\overline{WR}$)	33	I/O	G0 (Строб записи во внешнее ОЗУ)
PG1($\overline{RD}$)	34	I/O	G1 (Строб чтения из внешнего ОЗУ)
PG2 (ALE)	43	I/0	G2 (Строб адреса внешнего ОЗУ)
PG3 (TOSC2)	18	I/O	G3 (Вход для подключения резонатора к таймеру/счетчику T2)

Продолжение таблицы 2.10

Обозначение	Номер вывода	Тип вывода	Описание
PG4 (TOSC1)	19	I/O	G4 (Выход для подключения резонатора к таймеру/счетчику T2)
$\overline{\text{PEN}}$	1	I	Разрешение программирования
AREF	62	P	Вход опорного напряжения для АЦП
AGND	63	P	Аналоговый общий вывод
AV_{CC}	64	P	Вывод источника питания АЦП
GND	22, 53	P	Общий вывод
V_{CC}	21, 52	P	Вывод источника питания

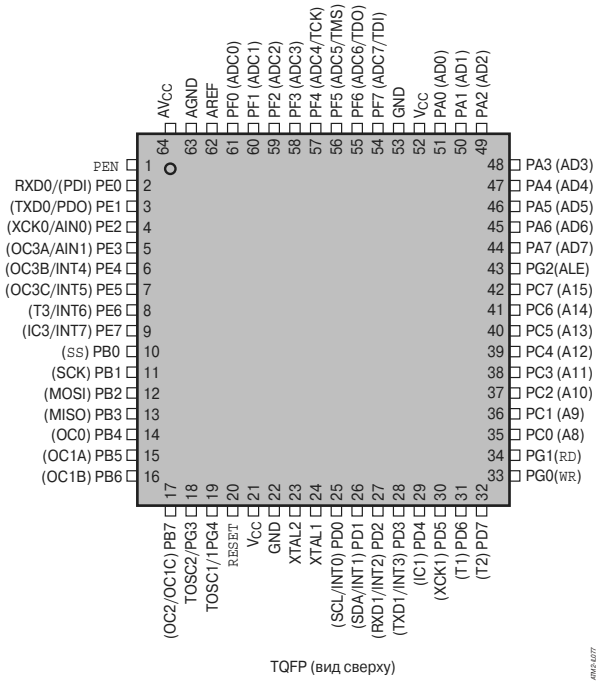


Рис. 2.10. Расположение выводов модели ATmega128(L)

Таблица 2.11. **Описание выводов модели ATmega128(L)**

Обозначение	Номер вывода	Тип вывода	Описание
XTAL1	24	I	Вход тактового генератора
XTAL2	23	O	Выход тактового генератора
$\overline{\text{RESET}}$	20	I	Вход сброса. При удержании на входе НИЗКОГО уровня в течение 50 нс выполняется сброс устройства
Порт A. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PA0 (AD0)	51	I/O	A0 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA1 (AD1)	50	I/O	A1 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA2 (AD2)	49	I/O	A2 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA3 (AD3)	48	I/O	A3 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA4 (AD4)	47	I/O	A4 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA5 (AD5)	46	I/O	A5 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA6 (AD6)	45	I/0	A6 (Мультиплексированная ША/ШД для внешнего ОЗУ)
PA7 (AD7)	44	I/O	A7 (Мультиплексированная ША/ШД для внешнего ОЗУ)
Порт B. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PB0 ($\overline{\text{SS}}$)	10	I/O	B0 (Выбор Slave-устройства на шине SPI)
PB1 (SCK)	11	I/O	B1 (Выход (Master) или вход (Slave) тактового сигнала модуля SPI)
PB2 (MOSI)	12	I/O	B2 (Выход (Master) или вход (Slave) данных модуля SPI)
PB3 (MISO)	13	I/O	B3 (Вход (Master) или выход (Slave) данных модуля SPI)
PB4 (OC0)	14	I/O	B4 (Выход таймера/счетчика T0 (режимы Compare, PWM))
PB5 (OC1A)	15	I/O	B5 (Выход A таймера/счетчика T1 (режимы Compare, PWM))
PB6(OC1B)	16	I/O	B6 (Выход B таймера/счетчика T1 (режимы Compare, PWM))
PB7 (OC2/OC1C)	17	I/O	B7 (Выход таймера/счетчика T2 (режимы Compare, PWM)/Выход C таймера/счетчика T1 (режимы Compare, PWM))
Порт C. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PC0 (A8)	35	I/O	C0 (ША для внешнего ОЗУ)
PC1 (A9)	36	I/O	C1 (ША для внешнего ОЗУ)

Продолжение таблицы 2.11

Обозначение	Номер вывода	Тип вывода	Описание
PC2 (A10)	37	I/O	C2 (ША для внешнего ОЗУ)
PC3 (A11)	38	I/O	C3 (ША для внешнего ОЗУ)
PC4 (A12)	39	I/O	C4 (ША для внешнего ОЗУ)
PC5 (A13)	40	I/O	C5 (ША для внешнего ОЗУ)
PC6 (A14)	41	I/O	C6 (ША для внешнего ОЗУ)
PC7 (A15)	42	I/O	C7 (ША для внешнего ОЗУ)
Порт D. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PD0 (SCL/INT0)	25	I/O	D0 (Тактовый сигнал модуля TWI/Вход внешнего прерывания)
PD1 (SDA/INT1)	26	I/O	D1 (Линия данных модуля TWI/Вход внешнего прерывания)
PD2 (RXD1/INT2)	27	I/O	D2 (Вход 2-го USART/Вход внешнего прерывания)
PD3 (TXD1/INT3)	28	I/O	D3 (Выход 2-го USART/Вход внешнего прерывания)
PD4 (ICP1)	29	I/O	D4 (Вход захвата таймера/счетчика T1 (режим Capture))
PD5 (XCK1)	30	I/O	D5 (Вход/выход внешнего тактового сигнала 2-го USART)
PD6 (T1)	31	I/O	D6 (Вход внешнего тактового сигнала таймера/счетчика T1)
PD7 (T2)	32	I/O	D7 (Вход внешнего тактового сигнала таймера/счетчика T2)
Порт E. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PE0 (RXD0/PDI)	2	I/O	E0 (Вход 1-го USART/Вход данных при последовательном программировании)
PE1 (TXD0/PDO)	3	I/O	E1 (Выход 1-го USART/Выход данных при последовательном программировании)
PE2 (XCK0/AIN0)	4	I/O	E2 (Вход/выход внешнего тактового сигнала 1-го USART/Положительный вход компаратора)
PE3 (OC3A/AIN1)	5	I/O	E3 (Выход A таймера/счетчика T3 (режимы Compare, PWM)/Отрицательный вход компаратора)
PE4 (OC3B/INT4)	6	I/O	E4 (Выход B таймера/счетчика T3 (режимы Compare, PWM)/Вход внешнего прерывания)
PE5 (OC3C/INT5)	7	I/O	E5 (Выход C таймера/счетчика T3 (режимы Compare, PWM)/Вход внешнего прерывания)
PE6 (T3/INT6)	8	I/O	E6 (Вход внешнего тактового сигнала таймера/счетчика T1/Вход внешнего прерывания)
PE7 (ICP3/INT7)	9	I/O	E7 (Вход захвата таймера/счетчика T1 (режим Capture)/Вход внешнего прерывания)

Продолжение таблицы 2.11

Обозначение	Номер вывода	Тип вывода	Описание
Порт F. 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PF0 (ADC0)	61	I/O	F0 (Вход АЦП)
PF1 (ADC1)	60	I/O	F1 (Вход АЦП)
PF2 (ADC2)	59	I/O	F2 (Вход АЦП)
PF3 (ADC3)	58	I/O	F3 (Вход АЦП)
PF4 (ADC4/TCK)	57	I/O	F4 (Вход АЦП/Тактовый сигнал JTAG)
PF5 (ADC5/TMS)	56	I/O	F5 Вход АЦП/Выбор режима JTAG)
PF6 (ADC6/TDO)	55	I/0	F6 (Вход АЦП/Выход данных JTAG)
PF7 (ADC7/TDI)	54	I/O	F7 (Вход АЦП/Вход данных JTAG)
Порт G. 5-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PG0($\overline{WR}$)	33	I/O	G0 (Строб записи во внешнее ОЗУ)
PG1($\overline{RD}$)	34	I/O	G1 (Строб чтения из внешнего ОЗУ)
PG2 (ALE)	43	I/0	G2 (Строб адреса внешнего ОЗУ)
PG3 (TOSC2)	18	I/O	G3 (Вход для подключения резонатора к таймеру/счетчику T2)
PG4 (TOSC1)	19	I/O	G4 (Выход для подключения резонатора к таймеру/счетчику T2)
$\overline{PEN}$	1	I	Разрешение программирования
AREF	62	P	Вход опорного напряжения для АЦП
AGND	63	P	Аналоговый общий вывод
AV_{CC}	64	P	Вывод источника питания АЦП
GND	22, 53	P	Общий вывод
V_{CC}	21, 52	P	Вывод источника питания

Глава 9. Архитектура микроконтроллеров семейства Mega

9.1. Введение

Микроконтроллеры AVR семейства Mega являются 8-разрядными микроконтроллерами с RISC-архитектурой. Они имеют электрически стираемую память программ (FLASH) и данных (EEPROM), а также разнообразные периферийные устройства. Следует отметить, что микроконтроллеры семейства Mega имеют самый богатый набор периферийных устройств по сравнению с микроконтроллерами других семейств. Более того, состав этих устройств от модели к модели практически не меняется (меняются только их функциональные возможности). К устройствам, присутствующим не во всех моделях семейства, относятся АЦП, модуль двухпроводного интерфейса TWI (Two Wire Interface, аналог шины I^2C), а также модуль интерфейса JTAG.

Структурная схема микроконтроллеров семейства Mega приведена на **Рис. 2.11**. Заметим, что на этом рисунке изображена структурная схема наиболее совершенного на сегодняшний день представителя семейства, ATmega128x. При рассмотрении других моделей, необходимо принимать во внимание присущие им ограничения, такие как наличие тех или иных периферийных устройств (см. Приложение 1) и использование контактов ввода/вывода этими устройствами (см. **Табл. 2.2...2.11**).

9.2. Организация памяти

В микроконтроллерах AVR семейства Mega реализована Гарвардская архитектура, в соответствии с которой разделены не только адресные пространства памяти программ и памяти данных, но также и шины доступа к ним. Способы адресации и доступа к этим областям памяти также различны. Такая структура позволяет центральному процессору работать одновременно как с памятью программ, так и с памятью данных, что существенно

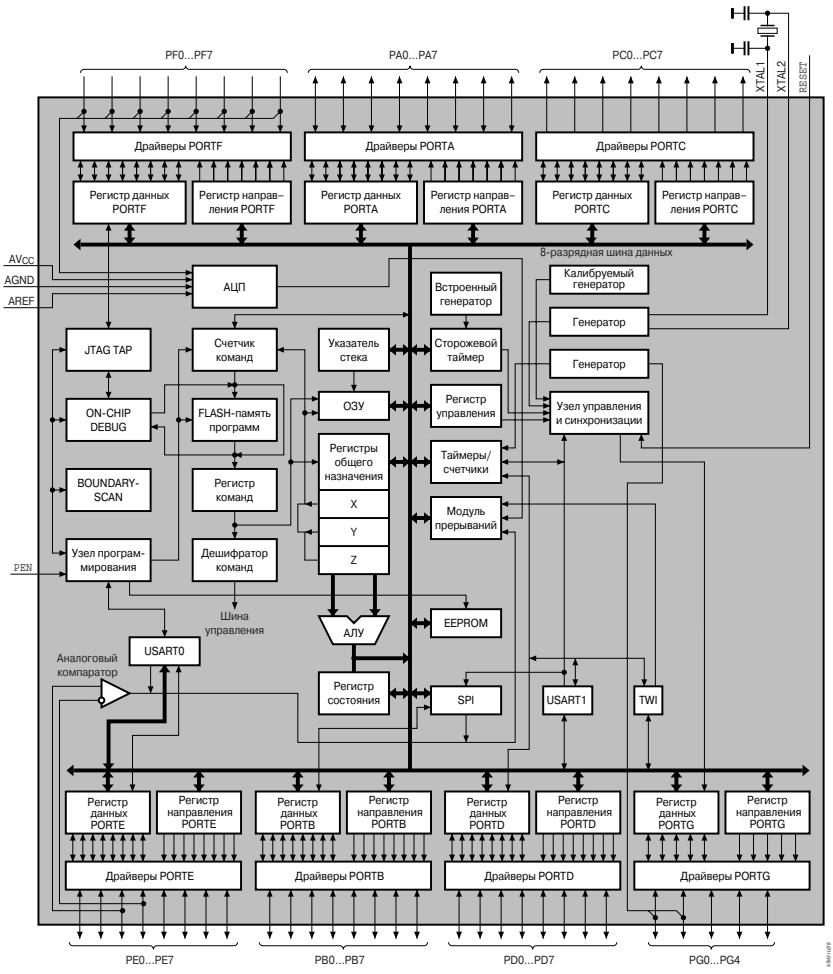
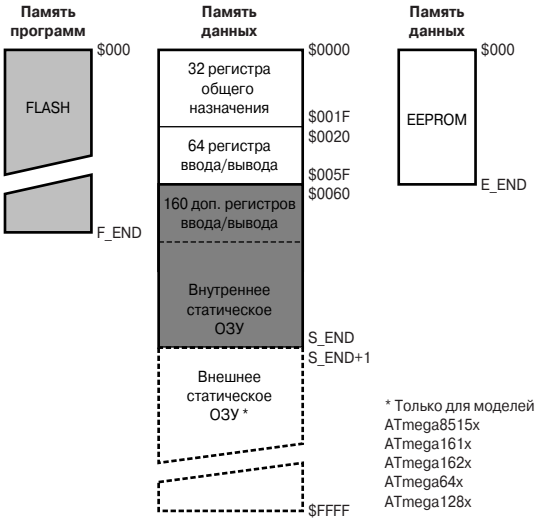


Рис. 2.11. Структурная схема микроконтроллеров семейства Mega

увеличивает производительность. Каждая из областей памяти данных (ОЗУ и EEPROM) также расположена в своем адресном пространстве.

Обобщенная карта памяти микроконтроллеров AVR семейства Mega приведена на **Рис. 2.12**.



Модель	Память программ (Flash)		Память данных (ОЗУ)		Память данных (EEPROM)	
	Верхняя граница (F_END)	Объем, слов	Верхняя граница (S_END)	Объем, Кбайт	Верхняя граница (E_END)	Объем, байт
ATmega8x	$0FFF	4 K	$045F	1	$1FF	512
ATmega8515x	$0FFF	4 K	$025F	0.512	$1FF	512
ATmega16x	$1FFF	8 K	$045F	1	$1FF	512
ATmega161x	$1FFF	8 K	$045F	1	$1FF	512
ATmega162x	$1FFF	8 K	$045F/$04FF	1	$1FF	512
ATmega163x	$1FFF	8 K	$045F	1	$1FF	512
ATmega32x	$3FFF	16 K	$085F	2	$3FF	1 K
ATmega323x	$3FFF	16 K	$085F	2	$3FF	1 K
ATmega64x	$7FFF	32 K	$10FF	4	$7FF	2 K
ATmega128x	$FFFF	64 K	$10FF	4	$FFF	4 K

Рис. 2.12. Карта памяти микроконтроллеров семейства Mega

9.2.1. Память программ

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

Как уже было сказано, 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-разрядную организацию. Соответственно, объем памяти микроконтроллеров семейства составляет от 4K (4×1024) до 64K (64×1024) 16-разрядных слов. Логически память программ 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. Подробно использование этой области и реализация программы-загрузчика будут рассмотрены в 4-й части книги (Глава 6). Если же возможность самопрограммирования микроконтроллера не используется, прикладная программа может располагаться и в области загрузчика.

Для адресации памяти программ используется счетчик команд (PC — Program Counter). Размер счетчика команд составляет 12...16 разрядов, 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 ATmega8x и ATmega8515x) или $0002 (остальные модели) памяти программ, располагается таблица векторов прерываний. Размер этой области зависит от модели микроконтроллера (подробнее о распределении области векторов прерывания см. раздел 4.2).

При возникновении прерывания после сохранения в стеке текущего значения счетчика команд происходит выполнение команды, расположенной по адресу соответствующего вектора. Поэтому по этим адресам располагаются команды перехода к подпрограммам обработки прерываний. В моделях с объемом памяти 8 Кбайт (ATmega8x и ATmega8515x) в качестве этих команд используются команды относительного перехода (RJMP), а в остальных моделях — команды абсолютного перехода (JMP).

В отличие от микроконтроллеров AVR других семейств в большинстве микроконтроллеров семейства Mega положение вектора сброса и/или таблицы векторов прерываний может быть изменено. Они могут располагаться не только в начале памяти программ, как описано выше, но и в начале области загрузчика. Подробнее об этом будет сказано в 4-й части книги (Глава 26).

Если прерывания в программе не используются либо таблица векторов прерываний располагается в области загрузчика, то основная программа может начинаться непосредственно с адреса $0001.

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 име-

ется две специальных команды — LPM и ELPM (последняя используется только в моделях ATmega128x). При использовании команды LPM адрес, по которому производится чтение, определяется содержимым индексного регистра Z (см. далее). При этом старшие 15 разрядов содержимого регистра будут определять адрес слова (0...32K), а младший разряд будет определять, какой из байтов будет прочитан: «0» — младший байт, «1» — старший байт (**Рис. 2.13, а**). Команда ELPM в отличие от команды LPM способна адресовать все 128 Кбайт памяти программ микроконтроллеров ATmega128x. При использовании этой команды адрес слова будет определяться разрядом RAMPZ0 регистра ввода/вывода RAMPZ совместно со старшими 15 разрядами содержимого регистра Z. Младший разряд регистра Z будет по-прежнему определять, какой из байтов слова будет прочитан (**Рис. 2.13, б**).

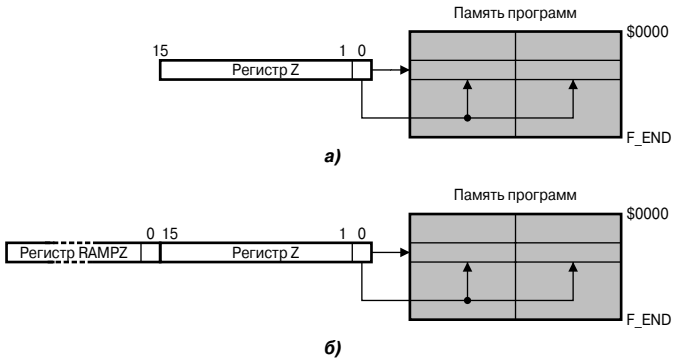


Рис. 2.13. Косвенная адресация памяти программ при использовании команды LPM *(а)* и команды ELPM *(б)*

Регистр RAMPZ расположен по адресу $3B ($5B) в основном пространстве РВВ микроконтроллеров ATmega128x, а его формат показан на **Рис. 2.14**.

	7	6	5	4	3	2	1	0
$35	-	-	-	-	-	-	-	RAMPZ0
Чтение (R)/Запись (W)	R	R	R	R	R	R	R	R/W
Начальное значение	0	0	0	0	0	0	0	0

Рис. 2.14. Формат регистра RAMPZ

В заключение следует отметить, что FLASH-ПЗУ, используемое в микроконтроллерах AVR, рассчитано как минимум на 1000 циклов стирания/записи.

9.2.2. 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 моделях ATmega162x, ATmega64x и ATmega128x имеется также область дополнительных (extended) регистров ввода/вывода (ДРВВ). Под РВВ в памяти микроконтроллера отводится 64 байт, а под ДРВВ — 160 байт.

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

Для хранения переменных программ помимо регистров общего назначения также может использоваться статическое ОЗУ объемом от 512 байт до 4 Кбайт. Ряд микроконтроллеров семейства, кроме того, имеют возможность подключения внешнего статического ОЗУ объемом до 64 Кбайт.

Для долговременного хранения различной информации, которая может изменяться в процессе функционирования готовой системы (калибровочные константы, серийные номера, ключи и т. п.), в микроконтроллерах семейства может использоваться EEPROM-память. Ее объем составляет для различных моделей 512 байт...4 Кбайт. Эта память расположена в отдельном адресном пространстве, а доступ к ней осуществляется с помощью определенных РВВ.

9.2.2.1. Статическое ОЗУ

Прежде всего, следует сказать, что в микроконтроллерах AVR семейства Mega используется линейная организация памяти. Объем статического ОЗУдля различных моделей семейства составляет 512 байт...4 Кбайт (см. **Табл. 2.12**).

В адресном пространстве ОЗУ также расположены все регистры микроконтроллеров, под них отведены младшие 96 (256) адресов (**Рис. 2.15**). Остальные адреса отведены под 512/1K/2K/4K...64K ячеек статического ОЗУ.

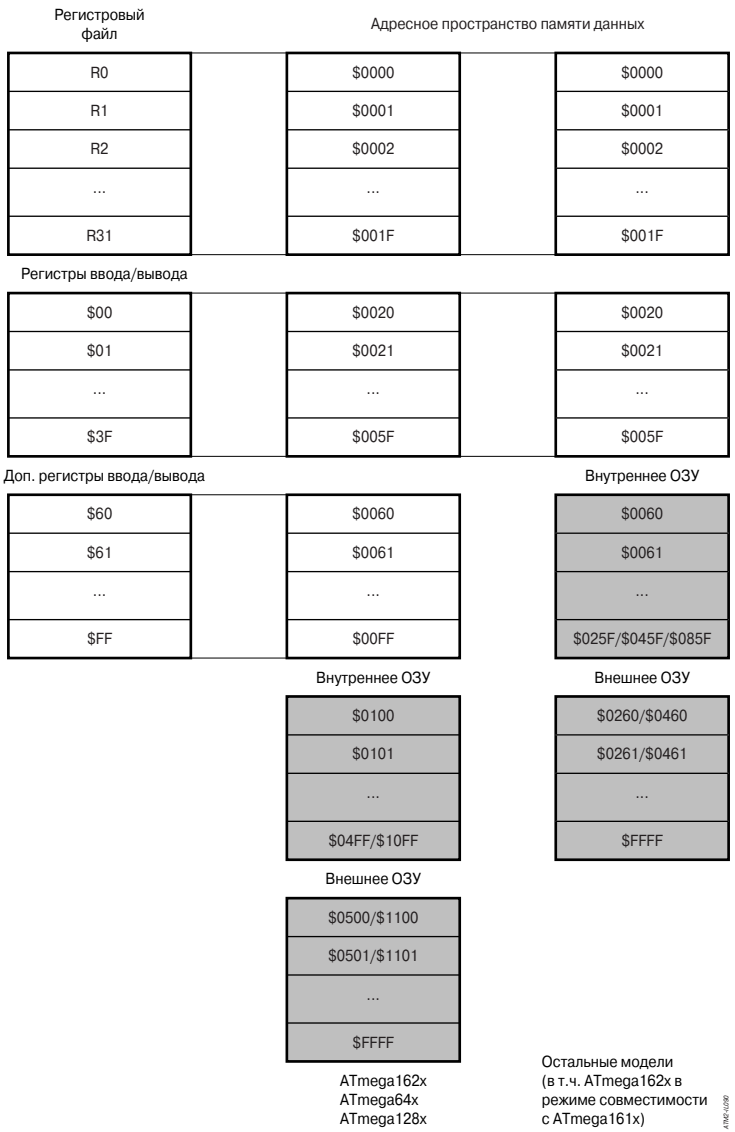


Рис. 2.15. Организация статического ОЗУ

9.2.2.2. Регистры общего назначения

Все регистры общего назначения объединены в регистровый файл быстрого доступа, структура которого показана на **Рис. 2.16**. Как уже было сказано, в микроконтроллерах AVR все 32 РОН непосредственно доступны АЛУ, в отличие от микроконтроллеров других фирм, в которых 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 решение (в сочетании с конвейерной обработкой) позволяет АЛУ выполнять одну операцию (извлечение операндов из регистрового файла, выполнение команды и запись результата обратно в регистровый файл) за один машинный цикл.

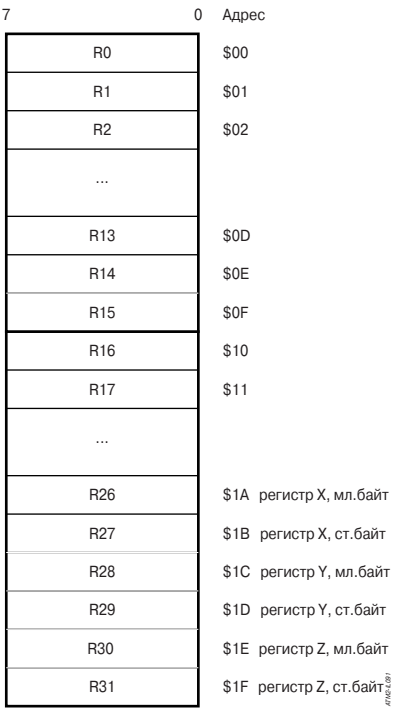


Рис. 2.16. Структура регистрового файла

Последние 6 регистров файла (R26...R31) могут также объединяться в три 16-разрядных регистра X, Y и Z (**Рис. 2.17**), используемых в качестве

указателей при косвенной адресации памяти данных. Подробно использование регистров-указателей описано в подразделе 9.2.2.5 «Способы адресации памяти данных».

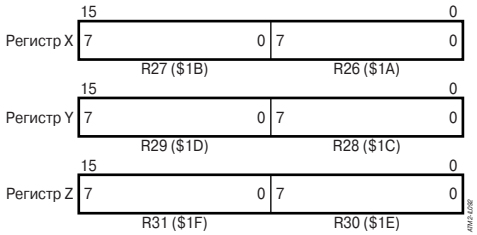


Рис. 2.17. Регистры-указатели X, Y и Z

Как показано на **Рис. 2.15** 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.

9.2.2.3. Регистры ввода/вывода

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

Во всех микроконтроллерах семейства Mega (как и в микроконтроллерах остальных семейств) регистры ввода/вывода располагаются в так называемом пространстве ввода/вывода размером 64 байта. В моделях ATmega162x, ATmega64x и ATmega128x, кроме того, имеется пространство дополнительных регистров ввода/вывода размером 160 байт. Введение дополнительных РВВ связано с тем, что для поддержки всех периферийных устройств этих моделей 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. 2.12...2.20**.

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 (для совместимости с будущими моделями) и запись по этому адресу запрещена.

Таблица 2.12. **Регистры ввода/вывода моделей ATmega8x**

Название	Адрес	Функция
SREG	$3F ($5F)	Регистр состояния
SPH	$3E ($5E)	Указатель стека, старший байт
SPL	$3D ($5D)	Указатель стека, младший байт
GICR	$3B ($5B)	Общий регистр управления прерываниями
GIFR	$3A ($5A)	Общий регистр флагов прерываний
TIMSK	$39 ($59)	Регистр маски прерываний от таймеров/счетчиков
TIFR	$38 ($58)	Регистр флагов прерываний от таймеров/счетчиков
SPMCR	$37 ($57)	Регистр управления памятью программ
TWCR	$36 ($56)	Регистр управления TWI
MCUCR	$35 ($55)	Регистр управления микроконтроллером
MCUCSR	$34 ($54)	Регистр управления и состояния микроконтроллера
TCCR0	$33 ($53)	Регистр управления таймером/счетчиком T0
TCNT0	$32 ($52)	Счетный регистр таймера/счетчика T0
OSCCAL	$31 ($51)	Регистр калибровки тактового генератора
SFIOR	$30 ($50)	Регистр специальных функций
TCCR1A	$2F ($4F)	Регистр управления A таймера/счетчика T1
TCCR1B	$2E ($4E)	Регистр управления B таймера/счетчика T1
TCNT1H	$2D ($4D)	Счетный регистр таймера/счетчика T1, старший байт
TCNT1L	$2C ($4C)	Счетный регистр таймера/счетчика T1, младший байт
OCR1AH	$2B ($4B)	Регистр совпадения A таймера/счетчика T1, старший байт
OCR1AL	$2A ($4A)	Регистр совпадения A таймера/счетчика T1, младший байт
OCR1BH	$29 ($49)	Регистр совпадения B таймера/счетчика T1, старший байт
OCR1BL	$28 ($48)	Регистр совпадения B таймера/счетчика T1, младший байт
ICR1H	$27 ($47)	Регистр захвата таймера/счетчика T1, старший байт
ICR1L	$26 ($46)	Регистр захвата таймера/счетчика T1, младший байт
TCCR2	$25 ($45)	Счетный регистр таймера/счетчика T2
TCNT2	$24 ($44)	Регистр совпадения таймера/счетчика T2
OCR2	$23 ($43)	Регистр совпадения таймера/счетчика T2
ASSR	$22 ($42)	Регистр состояния асинхронного режима
WDTCR	$21 ($41)	Регистр управления сторожевым таймером
UBRRH	$20 ($40)	Регистр скорости передачи USART, старший байт
UCSRC		Регистр управления и состояния USART
EEARH	$1F ($3F)	Регистр адреса EEPROM, старший байт
EEARL	$1E ($3E)	Регистр адреса EEPROM, младший байт
EEDR	$1D ($3D)	Регистр данных EEPROM
EECR	$1C ($3C)	Регистр управления EEPROM
PORTB	$18 ($38)	Регистр данных порта B
DDRB	$17 ($37)	Регистр направления данных порта B
PINB	$16 ($36)	Выводы порта B

Продолжение таблицы 2.12

Название	Адрес	Функция
PORTC	$15 ($35)	Регистр данных порта C
DDRC	$14 ($34)	Регистр направления данных порта C
PINC	$13 ($33)	Выводы порта C
PORTD	$12 ($32)	Регистр данных порта D
DDRD	$11 ($31)	Регистр направления данных порта D
PIND	$10 ($30)	Выводы порта D
SPDR	$0F ($2F)	Регистр данных SPI
SPSR	$0E ($2E)	Регистр состояния SPI
SPCR	$0D ($2D)	Регистр управления SPI
UDR	$0C ($2C)	Регистр данных USART
UCSRA	$0B ($2B)	Регистр управления и состояния A USART
UCSRB	$0A ($2A)	Регистр управления и состояния B USART
UBRRL	$09 ($29)	Регистр скорости передачи USART, младший байт
ACSR	$08 ($28)	Регистр управления и состояния аналогового компаратора
ADMUX	$07 ($27)	Регистр управления мультиплексором АЦП
ADCSR	$06 ($26)	Регистр управления и состояния АЦП
ADCH	$05 ($25)	Регистр данных АЦП, старший байт
ADCL	$04 ($24)	Регистр данных АЦП, младший байт
TWDR	$03 ($23)	Регистр данных TWI
TWAR	$02 ($22)	Регистр адреса TWI
TWSR	$01 ($21)	Регистр состояния TWI
TWBR	$00 ($20)	Регистр скорости передачи TWI

Таблица 2.13. **Регистры ввода/вывода моделей ATmega8515x**

Название	Адрес	Функция
SREG	$3F ($5F)	Регистр состояния
SPH	$3E ($5E)	Указатель стека, старший байт
SPL	$3D ($5D)	Указатель стека, младший байт
GICR	$3B ($5B)	Общий регистр управления прерываниями
GIFR	$3A ($5A)	Общий регистр флагов прерываний
TIMSK	$39 ($59)	Регистр маски прерываний от таймеров/счетчиков
TIFR	$38 ($58)	Регистр флагов прерываний от таймеров/счетчиков
SPMCR	$37 ($57)	Регистр управления памятью программ
EMCUCR	$36 ($56)	Дополнительный регистр управления микроконтроллером
MCUCR	$35 ($55)	Регистр управления микроконтроллером
MCUCSR	$34 ($54)	Регистр управления и состояния микроконтроллера
TCCR0	$33 ($53)	Регистр управления таймером/счетчиком T0
TCNT0	$32 ($52)	Счетный регистр таймера/счетчика T0
OCR0	$31 ($51)	Регистр совпадения таймера/счетчика T0
SFIOR	$30 ($50)	Регистр специальных функций
TCCR1A	$2F ($4F)	Регистр управления A таймера/счетчика T1

Продолжение таблицы 2.13

Название	Адрес	Функция
TCCR1B	$2E ($4E)	Регистр управления B таймера/счетчика T1
TCNT1H	$2D ($4D)	Счетный регистр таймера/счетчика T1, старший байт
TCNT1L	$2C ($4C)	Счетный регистр таймера/счетчика T1, младший байт
OCR1AH	$2B ($4B)	Регистр совпадения A таймера/счетчика T1, старший байт
OCR1AL	$2A ($4A)	Регистр совпадения A таймера/счетчика T1, младший байт
OCR1BH	$29 ($49)	Регистр совпадения B таймера/счетчика T1, старший байт
OCR1BL	$28 ($48)	Регистр совпадения B таймера/счетчика T1, младший байт
ICR1H	$25 ($45)	Регистр захвата таймера/счетчика T1, старший байт
ICR1L	$24 ($44)	Регистр захвата таймера/счетчика T1, младший байт
WDTCR	$21 ($41)	Регистр управления сторожевым таймером
UBRRH	$20 ($40)	Регистр скорости передачи USART, старший байт
UCSRC		Регистр управления и состояния USART
EEARH	$1F ($3F)	Регистр адреса EEPROM, старший байт
EEARL	$1E ($3E)	Регистр адреса EEPROM, младший байт
EEDR	$1D ($3D)	Регистр данных EEPROM
EECR	$1C ($3C)	Регистр управления EEPROM
PORTA	$1B ($3B)	Регистр данных порта A
DDRA	$1A ($3A)	Регистр направления данных порта A
PINA	$19 ($39)	Выводы порта A
PORTB	$18 ($38)	Регистр данных порта B
DDRB	$17 ($37)	Регистр направления данных порта B
PINB	$16 ($36)	Выводы порта B
PORTC	$15 ($35)	Регистр данных порта C
DDRC	$14 ($34)	Регистр направления данных порта C
PINC	$13 ($33)	Выводы порта C
PORTD	$12 ($32)	Регистр данных порта D
DDRD	$11 ($31)	Регистр направления данных порта D
PIND	$10 ($30)	Выводы порта D
SPDR	$0F ($2F)	Регистр данных SPI
SPSR	$0E ($2E)	Регистр состояния SPI
SPCR	$0D ($2D)	Регистр управления SPI
UDR	$0C ($2C)	Регистр данных USART
UCSRA	$0B ($2B)	Регистр управления и состояния A USART
UCSRB	$0A ($2A)	Регистр управления и состояния B USART
UBRRL	$09 ($29)	Регистр скорости передачи USART, младший байт
ACSR	$08 ($28)	Регистр управления и состояния аналогового компаратора
PORTE	$07 ($27)	Регистр данных порта E
DDRE	$06 ($26)	Регистр направления данных порта E
PINE	$05 ($25)	Выводы порта E
OSCCAL	$04 ($24)	Регистр калибровки тактового генератора

Таблица 2.14. **Регистры ввода/вывода моделей ATmega16x**

Название	Адрес	Функция
SREG	$3F ($5F)	Регистр состояния
SPH	$3E ($5E)	Указатель стека, старший байт
SPL	$3D ($5D)	Указатель стека, младший байт
OCR0	$3C ($5C)	Регистр совпадения таймера/счетчика T0
GICR	$3B ($5B)	Общий регистр управления прерываниями
GIFR	$3A ($5A)	Общий регистр флагов прерываний
TIMSK	$39 ($59)	Регистр маски прерываний от таймеров/счетчиков
TIFR	$38 ($58)	Регистр флагов прерываний от таймеров/счетчиков
SPMCR	$37 ($57)	Регистр управления памятью программ
TWCR	$36 ($56)	Регистр управления TWI
MCUCR	$35 ($55)	Регистр управления микроконтроллером
MCUCSR	$34 ($54)	Регистр управления и состояния микроконтроллера
TCCR0	$33 ($53)	Регистр управления таймером/счетчиком T0
TCNT0	$32 ($52)	Счетный регистр таймера/счетчика T0
OSCCAL	$31 ($51)	Регистр калибровки тактового генератора
OCDR		Регистр внутрисхемной отладки
SFIOR	$30 ($50)	Регистр специальных функций
TCCR1A	$2F ($4F)	Регистр управления A таймера/счетчика T1
TCCR1B	$2E ($4E)	Регистр управления B таймера/счетчика T1
TCNT1H	$2D ($4D)	Счетный регистр таймера/счетчика T1, старший байт
TCNT1L	$2C ($4C)	Счетный регистр таймера/счетчика T1, младший байт
OCR1AH	$2B ($4B)	Регистр совпадения A таймера/счетчика T1, старший байт
OCR1AL	$2A ($4A)	Регистр совпадения A таймера/счетчика T1, младший байт
OCR1BH	$29 ($49)	Регистр совпадения B таймера/счетчика T1, старший байт
OCR1BL	$28 ($48)	Регистр совпадения B таймера/счетчика T1, младший байт
ICR1H	$27 ($47)	Регистр захвата таймера/счетчика T1, старший байт
ICR1L	$26 ($46)	Регистр захвата таймера/счетчика T1, младший байт
TCCR2	$25 ($45)	Счетный регистр таймера/счетчика T2
TCNT2	$24 ($44)	Регистр совпадения таймера/счетчика T2
OCR2	$23 ($43)	Регистр совпадения таймера/счетчика T2
ASSR	$22 ($42)	Регистр состояния асинхронного режима
WDTCR	$21 ($41)	Регистр управления сторожевым таймером
UBRRH	$20 ($40)	Регистр скорости передачи USART, старший байт
UCSRC		Регистр управления и состояния USART
EEARH	$1F ($3F)	Регистр адреса EEPROM, старший байт
EEARL	$1E ($3E)	Регистр адреса EEPROM, младший байт
EEDR	$1D ($3D)	Регистр данных EEPROM
EECR	$1C ($3C)	Регистр управления EEPROM
PORTA	$1B ($3B)	Регистр данных порта A

Продолжение таблицы 2.14

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

Таблица 2.15. **Регистры ввода/вывода моделей ATmega161x**

Название	Адрес	Функция
SREG	$3F ($5F)	Регистр состояния
SPH	$3E ($5E)	Указатель стека, старший байт
SPL	$3D ($5D)	Указатель стека, младший байт
GIMSK	$3B ($5B)	Общий регистр маски прерываний
GIFR	$3A ($5A)	Общий регистр флагов прерываний
TIMSK	$39 ($59)	Регистр маски прерываний от таймеров/счетчиков
TIFR	$38 ($58)	Регистр флагов прерываний от таймеров/счетчиков
SPMCR	$37 ($57)	Регистр управления памятью программ

Продолжение таблицы 2.15

Название	Адрес	Функция
EMCUCR	$36 ($56)	Дополнительный регистр управления микроконтроллером
MCUCR	$35 ($55)	Регистр управления микроконтроллером
MCUSR	$34 ($54)	Регистр состояния микроконтроллера
TCCR0	$33 ($53)	Регистр управления таймером/счетчиком T0
TCNT0	$32 ($52)	Счетный регистр таймера/счетчика T0
OCR0	$31 ($51)	Регистр совпадения таймера/счетчика T0
SFIOR	$30 ($50)	Регистр специальных функций
TCCR1A	$2F ($4F)	Регистр управления A таймера/счетчика T1
TCCR1B	$2E ($4E)	Регистр управления B таймера/счетчика T1
TCNT1H	$2D ($4D)	Счетный регистр таймера/счетчика T1, старший байт
TCNT1L	$2C ($4C)	Счетный регистр таймера/счетчика T1, младший байт
OCR1AH	$2B ($4B)	Регистр совпадения A таймера/счетчика T1, старший байт
OCR1AL	$2A ($4A)	Регистр совпадения A таймера/счетчика T1, младший байт
OCR1BH	$29 ($49)	Регистр совпадения B таймера/счетчика T1, старший байт
OCR1BL	$28 ($48)	Регистр совпадения B таймера/счетчика T1, младший байт
TCCR2	$27 ($47)	Регистр управления таймером/счетчиком T2
ASSR	$26 ($46)	Регистр состояния асинхронного режима
ICR1H	$25 ($45)	Регистр захвата таймера/счетчика T1, старший байт
ICR1L	$24 ($44)	Регистр захвата таймера/счетчика T1, младший байт
TCNT2	$23 ($43)	Счетный регистр таймера/счетчика T2
OCR2	$22 ($42)	Регистр совпадения таймера/счетчика T2
WDTCR	$21 ($41)	Регистр управления сторожевым таймером
UBRRHI	$20 ($40)	Регистр скорости передачи UART, старший байт
EEARH	$1F ($3F)	Регистр адреса EEPROM, старший байт
EEARL	$1E ($3E)	Регистр адреса EEPROM, младший байт
EEDR	$1D ($3D)	Регистр данных EEPROM
EECR	$1C ($3C)	Регистр управления EEPROM
PORTA	$1B($3B)	Регистр данных порта A
DDRA	$1A ($3A)	Регистр направления данных порта A
PINA	$19 ($39)	Выводы порта A
PORTB	$18 ($38)	Регистр данных порта B
DDRB	$17 ($37)	Регистр направления данных порта B
PINB	$16 ($36)	Выводы порта B
PORTC	$15 ($35)	Регистр данных порта C
DDRC	$14 ($34)	Регистр направления данных порта C
PINC	$13 ($33)	Выводы порта C
PORTD	$12 ($32)	Регистр данных порта D
DDRD	$11 ($31)	Регистр направления данных порта D

Продолжение таблицы 2.15

Название	Адрес	Функция
PIND	$10 ($30)	Выводы порта D
SPDR	$0F ($2F)	Регистр данных SPI
SPSR	$0E ($2E)	Регистр состояния SPI
SPCR	$0D ($2D)	Регистр управления SPI
UDR0	$0C ($2C)	Регистр данных UART0
UCSR0A	$0B ($2B)	Регистр управления и состояния A UART0
UCSR0B	$0A ($2A)	Регистр управления и состояния B UART0
UBRR0	$09 ($29)	Регистр скорости передачи UART0
ACSR	$08 ($28)	Регистр управления и состояния аналогового компаратора
PORTE	$07 ($27)	Регистр данных порта E
DDRE	$06 ($26)	Регистр направления данных порта E
PINE	$05 ($25)	Выводы порта E
UDR1	$03 ($23)	Регистр данных UART1
UCSR1A	$02 ($22)	Регистр управления и состояния A UART1
UCSR1B	$01 ($21)	Регистр управления и состояния B UART1
UBRR1	$00 ($20)	Регистр скорости передачи UART1

Таблица 2.16. **Регистры ввода/вывода моделей ATmega162x**

Название	Адрес	Функция
TCCR3A	($8B)	Регистр управления A таймера/счетчика T3
TCCR3B	($8A)	Регистр управления B таймера/счетчика T3
TCNT3H	($89)	Счетный регистр таймера/счетчика T3, старший байт
TCNT3L	($88)	Счетный регистр таймера/счетчика T3, младший байт
OCR3AH	($87)	Регистр совпадения A таймера/счетчика T3, старший байт
OCR3AL	($86)	Регистр совпадения A таймера/счетчика T3, младший байт
OCR3BH	($85)	Регистр совпадения B таймера/счетчика T3, старший байт
OCR3BL	($84)	Регистр совпадения B таймера/счетчика T3, младший байт
ICR3H	($81)	Регистр захвата таймера/счетчика T3, старший байт
ICR3L	($80)	Регистр захвата таймера/счетчика T3, младший байт
ETIMSK	($7D)	Дополнительный регистр маски прерываний от таймеров/счетчиков
ETIFR	($7C)	Дополнительный регистр флагов прерываний от таймеров/счетчиков
PCMSK1	($6C)	Регистр маски 0 прерываний по изменению сигнала
PCMSK0	($6B)	Регистр маски 1 прерываний по изменению сигнала
CLKPCE	($61)	Регистр предделителя тактового сигнала
SREG	$3F ($5F)	Регистр состояния
SPH	$3E ($5E)	Указатель стека, старший байт
SPL	$3D ($5D)	Указатель стека, младший байт

Продолжение таблицы 2.16

Название	Адрес	Функция
UBRR1H	$3C ($5C)	Регистр скорости передачи USART1, старший байт
UCSR1C		Регистр управления и состояния C USART1
GICR	$3B ($5B)	Общий регистр управления прерываниями
GIFR	$3A ($5A)	Общий регистр флагов прерываний
TIMSK	$39 ($59)	Регистр маски прерываний от таймеров/счетчиков
TIFR	$38 ($58)	Регистр флагов прерываний от таймеров/счетчиков
SPMCR	$37 ($57)	Регистр управления памятью программ
EMCUCR	$36 ($56)	Дополнительный регистр управления микроконтроллером
MCUCR	$35 ($55)	Регистр управления микроконтроллером
MCUSR	$34 ($54)	Регистр состояния микроконтроллера
TCCR0	$33 ($53)	Регистр управления таймером/счетчиком T0
TCNT0	$32 ($52)	Счетный регистр таймера/счетчика T0
OCR0	$31 ($51)	Регистр совпадения таймера/счетчика T0
SFIOR	$30 ($50)	Регистр специальных функций
TCCR1A	$2F ($4F)	Регистр управления A таймера/счетчика T1
TCCR1B	$2E ($4E)	Регистр управления B таймера/счетчика T1
TCNT1H	$2D ($4D)	Счетный регистр таймера/счетчика T1, старший байт
TCNT1L	$2C ($4C)	Счетный регистр таймера/счетчика T1, младший байт
OCR1AH	$2B ($4B)	Регистр совпадения A таймера/счетчика T1, старший байт
OCR1AL	$2A ($4A)	Регистр совпадения A таймера/счетчика T1, младший байт
OCR1BH	$29 ($49)	Регистр совпадения B таймера/счетчика T1, старший байт
OCR1BL	$28 ($48)	Регистр совпадения B таймера/счетчика T1, младший байт
TCCR2	$27 ($47)	Регистр управления таймером/счетчиком T2
ASSR	$26 ($46)	Регистр состояния асинхронного режима
ICR1H	$25 ($45)	Регистр захвата таймера/счетчика T1, старший байт
ICR1L	$24 ($44)	Регистр захвата таймера/счетчика T1, младший байт
TCNT2	$23 ($43)	Счетный регистр таймера/счетчика T2
OCR2	$22 ($42)	Регистр совпадения таймера/счетчика T2
WDTCR	$21 ($41)	Регистр управления сторожевым таймером
UBRR0H	$20 ($40)	Регистр скорости передачи USART0, старший байт
UCSR0C		Регистр управления и состояния C USART0
EEARH	$1F ($3F)	Регистр адреса EEPROM, старший байт
EEARL	$1E ($3E)	Регистр адреса EEPROM, младший байт
EEDR	$1D ($3D)	Регистр данных EEPROM
EECR	$1C ($3C)	Регистр управления EEPROM
PORTA	$1B($3B)	Регистр данных порта A
DDRA	$1A ($3A)	Регистр направления данных порта A
PINA	$19 ($39)	Выводы порта A
PORTB	$18 ($38)	Регистр данных порта B
DDRB	$17 ($37)	Регистр направления данных порта B

Продолжение таблицы 2.16

Название	Адрес	Функция
PINB	$16 ($36)	Выводы порта B
PORTC	$15 ($35)	Регистр данных порта C
DDRC	$14 ($34)	Регистр направления данных порта C
PINC	$13 ($33)	Выводы порта C
PORTD	$12 ($32)	Регистр данных порта D
DDRD	$11 ($31)	Регистр направления данных порта D
PIND	$10 ($30)	Выводы порта D
SPDR	$0F ($2F)	Регистр данных SPI
SPSR	$0E ($2E)	Регистр состояния SPI
SPCR	$0D ($2D)	Регистр управления SPI
UDR0	$0C ($2C)	Регистр данных USART0
UCSR0A	$0B ($2B)	Регистр управления и состояния A USART0
UCSR0B	$0A ($2A)	Регистр управления и состояния B USART0
UBRR0L	$09 ($29)	Регистр скорости передачи USART0, младший байт
ACSR	$08 ($28)	Регистр управления и состояния аналогового компаратора
PORTE	$07 ($27)	Регистр данных порта E
DDRE	$06 ($26)	Регистр направления данных порта E
PINE	$05 ($25)	Выводы порта E
OSCCAL	$04 ($24)	Регистр калибровки тактового генератора
OCDR		Регистр внутрисхемной отладки
UDR1	$03 ($23)	Регистр данных USART1
UCSR1A	$02 ($22)	Регистр управления и состояния A USART1
UCSR1B	$01 ($21)	Регистр управления и состояния B USART1
UBRR1L	$00 ($20)	Регистр скорости передачи USART1, младший байт

Таблица 2.17. **Регистры вода/вывода моделей ATmega163x**

Название	Адрес	Функция
SREG	$3F ($5F)	Регистр состояния
SPH	$3E ($5E)	Указатель стека, старший байт
SPL	$3D ($5D)	Указатель стека, младший байт
GIMSK	$3B ($5B)	Общий регистр маски прерываний
GIFR	$3A ($5A)	Общий регистр флагов прерываний
TIMSK	$39 ($59)	Регистр маски прерываний от таймеров/счетчиков
TIFR	$38 ($58)	Регистр флагов прерываний от таймеров/счетчиков
SPMCR	$37 ($57)	Регистр управления памятью программ
TWCR	$36 ($56)	Регистр управления TWI
MCUCR	$35 ($55)	Регистр управления микроконтроллером
MCUSR	$34 ($54)	Регистр состояния микроконтроллера
TCCR0	$33 ($53)	Регистр управления таймером/счетчиком T0
TCNT0	$32 ($52)	Счетный регистр таймера/счетчика T0

Продолжение таблицы 2.17

Название	Адрес	Функция
OSCCAL	$31 ($51)	Регистр калибровки тактового генератора
SFIOR	$30 ($50)	Регистр специальных функций
TCCR1A	$2F ($4F)	Регистр управления A таймера/счетчика T1
TCCR1B	$2E ($4E)	Регистр управления B таймера/счетчика T1
TCNT1H	$2D ($4D)	Счетный регистр таймера/счетчика T1, старший байт
TCNT1L	$2C ($4C)	Счетный регистр таймера/счетчика T1, младший байт
OCR1AH	$2B ($4B)	Регистр совпадения A таймера/счетчика T1, старший байт
OCR1AL	$2A ($4A)	Регистр совпадения A таймера/счетчика T1, младший байт
OCR1BH	$29 ($49)	Регистр совпадения B таймера/счетчика T1, старший байт
OCR1BL	$28 ($48)	Регистр совпадения B таймера/счетчика T1, младший байт
ICR1H	$27 ($47)	Регистр захвата таймера/счетчика T1, старший байт
ICR1L	$26 ($46)	Регистр захвата таймера/счетчика T1, младший байт
TCCR2	$25 ($45)	Счетный регистр таймера/счетчика T2
TCNT2	$24 ($44)	Регистр совпадения таймера/счетчика T2
OCR2	$23 ($43)	Регистр совпадения таймера/счетчика T2
ASSR	$22 ($42)	Регистр состояния асинхронного режима
WDTCR	$21 ($41)	Регистр управления сторожевым таймером
UBRRHI	$20 ($40)	Регистр скорости передачи UART, старший байт
EEARH	$1F ($3F)	Регистр адреса EEPROM, старший байт
EEARL	$1E ($3E)	Регистр адреса EEPROM, младший байт
EEDR	$1D ($3D)	Регистр данных EEPROM
EECR	$1C ($3C)	Регистр управления EEPROM
PORTA	$1B ($3B)	Регистр данных порта A
DDRA	$1A ($3A)	Регистр направления данных порта A
PINA	$19 ($39)	Выводы порта A
PORTB	$18 ($38)	Регистр данных порта B
DDRB	$17 ($37)	Регистр направления данных порта B
PINB	$16 ($36)	Выводы порта B
PORTC	$15 ($35)	Регистр данных порта C
DDRC	$14 ($34)	Регистр направления данных порта C
PINC	$13 ($33)	Выводы порта C
PORTD	$12 ($32)	Регистр данных порта D
DDRD	$11 ($31)	Регистр направления данных порта D
PIND	$10 ($30)	Выводы порта D
SPDR	$0F ($2F)	Регистр данных SPI
SPSR	$0E ($2E)	Регистр состояния SPI
SPCR	$0D ($2D)	Регистр управления SPI
UDR	$0C ($2C)	Регистр данных UART
UCSRA	$0B ($2B)	Регистр управления и состояния A UART
UCSRB	$0A ($2A)	Регистр управления и состояния B UART
UBRR	$09 ($29)	Регистр скорости передачи UART

Продолжение таблицы 2.17

Название	Адрес	Функция
ACSR	$08 ($28)	Регистр управления и состояния аналогового компаратора
ADMUX	$07 ($27)	Регистр управления мультиплексором АЦП
ADCSR	$06 ($26)	Регистр управления и состояния АЦП
ADCH	$05 ($25)	Регистр данных АЦП, старший байт
ADCL	$04 ($24)	Регистр данных АЦП, младший байт
TWDR	$03 ($23)	Регистр данных TWI
TWAR	$02 ($22)	Регистр адреса TWI
TWSR	$01 ($21)	Регистр состояния TWI
TWBR	$00 ($20)	Регистр скорости передачи TWI

Таблица 2.18. **Регистры вода/вывода моделей ATmega32x**

Название	Адрес	Функция
SREG	$3F ($5F)	Регистр состояния
SPH	$3E ($5E)	Указатель стека, старший байт
SPL	$3D ($5D)	Указатель стека, младший байт
OCR0	$3C ($3C)	Регистр совпадения таймера/счетчика T0
GICR	$3B ($5B)	Общий регистр управления прерываниями
GIFR	$3A ($5A)	Общий регистр флагов прерываний
TIMSK	$39 ($59)	Регистр маски прерываний от таймеров/счетчиков
TIFR	$38 ($58)	Регистр флагов прерываний от таймеров/счетчиков
SPMCR	$37 ($57)	Регистр управления памятью программ
TWCR	$36 ($56)	Регистр управления TWI
MCUCR	$35 ($55)	Регистр управления микроконтроллером
MCUCSR	$34 ($54)	Регистр управления и состояния микроконтроллера
TCCR0	$33 ($53)	Регистр управления таймером/счетчиком T0
TCNT0	$32 ($52)	Счетный регистр таймера/счетчика T0
OSCCAL	$31 ($51)	Регистр калибровки тактового генератора
OCDR		Регистр внутрисхемной отладки
SFIOR	$30 ($50)	Регистр специальных функций
TCCR1A	$2F ($4F)	Регистр управления A таймера/счетчика T1
TCCR1B	$2E ($4E)	Регистр управления B таймера/счетчика T1
TCNT1H	$2D ($4D)	Счетный регистр таймера/счетчика T1, старший байт
TCNT1L	$2C ($4C)	Счетный регистр таймера/счетчика T1, младший байт
OCR1AH	$2B ($4B)	Регистр совпадения A таймера/счетчика T1, старший байт
OCR1AL	$2A ($4A)	Регистр совпадения A таймера/счетчика T1, младший байт
OCR1BH	$29 ($49)	Регистр совпадения B таймера/счетчика T1, старший байт
OCR1BL	$28 ($48)	Регистр совпадения B таймера/счетчика T1, младший байт
ICR1H	$27 ($47)	Регистр захвата таймера/счетчика T1, старший байт
ICR1L	$26 ($46)	Регистр захвата таймера/счетчика T1, младший байт

Продолжение таблицы 2.18

Название	Адрес	Функция
TCCR2	$25 ($45)	Счетный регистр таймера/счетчика T2
TCNT2	$24 ($44)	Регистр совпадения таймера/счетчика T2
OCR2	$23 ($43)	Регистр совпадения таймера/счетчика T2
ASSR	$22 ($42)	Регистр состояния асинхронного режима
WDTCR	$21 ($41)	Регистр управления сторожевым таймером
UBRRH	$20 ($40)	Регистр скорости передачи USART, старший байт
UCSRC		Регистр управления и состояния USART
EEARH	$1F ($3F)	Регистр адреса EEPROM, старший байт
EEARL	$1E ($3E)	Регистр адреса EEPROM, младший байт
EEDR	$1D ($3D)	Регистр данных EEPROM
EECR	$1C ($3C)	Регистр управления EEPROM
PORTA	$1B ($3B)	Регистр данных порта A
DDRA	$1A ($3A)	Регистр направления данных порта A
PINA	$19 ($39)	Выводы порта A
PORTB	$18 ($38)	Регистр данных порта B
DDRB	$17 ($37)	Регистр направления данных порта B
PINB	$16 ($36)	Выводы порта B
PORTC	$15 ($35)	Регистр данных порта C
DDRC	$14 ($34)	Регистр направления данных порта C
PINC	$13 ($33)	Выводы порта C
PORTD	$12 ($32)	Регистр данных порта D
DDRD	$11 ($31)	Регистр направления данных порта D
PIND	$10 ($30)	Выводы порта D
SPDR	$0F ($2F)	Регистр данных SPI
SPSR	$0E ($2E)	Регистр состояния SPI
SPCR	$0D ($2D)	Регистр управления SPI
UDR	$0C ($2C)	Регистр данных USART
UCSRA	$0B ($2B)	Регистр управления и состояния A USART
UCSRB	$0A ($2A)	Регистр управления и состояния B USART
UBRRL	$09 ($29)	Регистр скорости передачи USART, младший байт
ACSR	$08 ($28)	Регистр управления и состояния аналогового компаратора
ADMUX	$07 ($27)	Регистр управления мультиплексором АЦП
ADCSRA	$06 ($26)	Регистр управления и состояния АЦП
ADCH	$05 ($25)	Регистр данных АЦП, старший байт
ADCL	$04 ($24)	Регистр данных АЦП, младший байт
TWDR	$03 ($23)	Регистр данных TWI
TWAR	$02 ($22)	Регистр адреса TWI
TWSR	$01 ($21)	Регистр состояния TWI
TWBR	$00 ($20)	Регистр скорости передачи TWI

Таблица 2.19. **Регистры ввода/вывода моделей ATmega323x**

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

Продолжение таблицы 2.19

Название	Адрес	Функция
UBRRH	$20 ($40)	Регистр скорости передачи USART, старший байт
UCSRC		Регистр управления и состояния USART
EEARH	$1F ($3F)	Регистр адреса EEPROM, старший байт
EEARL	$1E ($3E)	Регистр адреса EEPROM, младший байт
EEDR	$1D ($3D)	Регистр данных EEPROM
EECR	$1C ($3C)	Регистр управления EEPROM
PORTA	$1B ($3B)	Регистр данных порта A
DDRA	$1A ($3A)	Регистр направления данных порта A
PINA	$19 ($39)	Выводы порта A
PORTB	$18 ($38)	Регистр данных порта B
DDRB	$17 ($37)	Регистр направления данных порта B
PINB	$16 ($36)	Выводы порта B
PORTC	$15 ($35)	Регистр данных порта C
DDRC	$14 ($34)	Регистр направления данных порта C
PINC	$13 ($33)	Выводы порта C
PORTD	$12 ($32)	Регистр данных порта D
DDRD	$11 ($31)	Регистр направления данных порта D
PIND	$10 ($30)	Выводы порта D
SPDR	$0F ($2F)	Регистр данных SPI
SPSR	$0E ($2E)	Регистр состояния SPI
SPCR	$0D ($2D)	Регистр управления SPI
UDR	$0C ($2C)	Регистр данных USART
UCSRA	$0B ($2B)	Регистр управления и состояния A USART
UCSRB	$0A ($2A)	Регистр управления и состояния B USART
UBRRL	$09 ($29)	Регистр скорости передачи USART, младший байт
ACSR	$08 ($28)	Регистр управления и состояния аналогового компаратора
ADMUX	$07 ($27)	Регистр управления мультиплексором АЦП
ADCSR	$06 ($26)	Регистр управления и состояния АЦП
ADCH	$05 ($25)	Регистр данных АЦП, старший байт
ADCL	$04 ($24)	Регистр данных АЦП, младший байт
TWDR	$03 ($23)	Регистр данных TWI
TWAR	$02 ($22)	Регистр адреса TWI
TWSR	$01 ($21)	Регистр состояния TWI
TWBR	$00 ($20)	Регистр скорости передачи TWI

Таблица 2.20. **Регистры ввода/вывода моделей ATmega64x/128x**

Название	Адрес	Функция
UCSR1C	($9D)	Регистр управления и состояния C USART1
UDR1	($9C)	Регистр данных USART1
UCSR1A	($9B)	Регистр управления и состояния A USART1
UCSR1B	($9A)	Регистр управления и состояния B USART1
UBRR1L	($99)	Регистр скорости передачи USART1, младший байт
UBRR1H	($98)	Регистр скорости передачи USART1, старший байт
UCSR0C	($95)	Регистр управления и состояния C USART0
UBRR0H	($90)	Регистр скорости передачи USART0, старший байт
ADCSRB	($8E) ATmega64	Регистр управления и состояния B АЦП
TCCR3C	($8C)	Регистр управления C таймера/счетчика T3
TCCR3A	($8B)	Регистр управления A таймера/счетчика T3
TCCR3B	($8A)	Регистр управления B таймера/счетчика T3
TCNT3H	($89)	Счетный регистр таймера/счетчика T3, старший байт
TCNT3L	($88)	Счетный регистр таймера/счетчика T3, младший байт
OCR3AH	($87)	Регистр совпадения A таймера/счетчика T3, старший байт
OCR3AL	($86)	Регистр совпадения A таймера/счетчика T3, младший байт
OCR3BH	($85)	Регистр совпадения B таймера/счетчика T3, старший байт
OCR3BL	($84)	Регистр совпадения B таймера/счетчика T31, младший байт
OCR3CH	($83)	Регистр совпадения C таймера/счетчика T3, старший байт
OCR3CL	($82)	Регистр совпадения C таймера/счетчика T3, младший байт
ICR3H	($81)	Регистр захвата таймера/счетчика T3, старший байт
ICR3L	($80)	Регистр захвата таймера/счетчика T3, младший байт
ETIMSK	($7D)	Дополнительный регистр маски прерываний от таймеров/счетчиков
ETIFR	($7C)	Дополнительный регистр флагов прерываний от таймеров/счетчиков
TCCR1C	($7A)	Регистр управления C таймера/счетчика T1
OCR1CH	($79)	Регистр совпадения C таймера/счетчика T1, старший байт
OCR1CL	($78)	Регистр совпадения C таймера/счетчика T1, младший байт
TWCR	($74)	Регистр управления TWI
TWDR	($73)	Регистр данных TWI
TWAR	($72)	Регистр адреса TWI
TWSR	($71	Регистр состояния TWI
TWBR	($70)	Регистр скорости передачи TWI
OSCCAL	($6F)	Регистр калибровки тактового генератора
XMCRA	($6D)	Регистр управления A внешней памятью
XMCRB	($6C)	Регистр управления B внешней памятью

Продолжение таблицы 2.20

Название	Адрес	Функция
EICRA	($6A)	Регистр управления A внешними прерываниями
SPMCR	($68)	Регистр управления памятью программ
PORTG	($65)	Регистр данных порта G
DDRG	($64)	Регистр направления данных порта G
PING	($63)	Выводы порта G
PORTF	($62)	Регистр данных порта F
DDRF	($61)	Регистр направления данных порта F
SREG	$3F ($5F)	Регистр состояния
SPH	$3E ($5E)	Указатель стека, старший байт
SPL	$3D ($5D)	Указатель стека, младший байт
XDIV	$3C ($5C)	Регистр управления делителем тактовой частоты
RAMPZ	$3B ($5B) ATmega128	Регистр выбора страницы
EICRB	$3A ($5A)	Регистр управления B внешними прерываниями
EIMSK	$39 ($59)	Регистр маски внешних прерываний
EIFR	$38 ($58)	Регистр флагов внешних прерываний
TIMSK	$37 ($57)	Регистр маски прерываний от таймеров/счетчиков
TIFR	$36 ($56)	Регистр флагов прерываний от таймеров/счетчиков
MCUCR	$35 ($55)	Регистр управления микроконтроллером
MCUCSR	$34 ($54)	Регистр управления и состояния микроконтроллера
TCCR0	$33 ($53)	Регистр управления таймером/счетчиком T0
TCNT0	$32 ($52)	Счетный регистр таймера/счетчика T0
OCR0	$31 ($51)	Регистр совпадения таймера/счетчика T0
ASSR	$30 ($50)	Регистр состояния асинхронного режима
TCCR1A	$2F ($4F)	Регистр управления A таймера/счетчика T1
TCCR1B	$2E ($4E)	Регистр управления B таймера/счетчика T1
TCNT1H	$2D ($4D)	Счетный регистр таймера/счетчика T1, старший байт
TCNT1L	$2C ($4C)	Счетный регистр таймера/счетчика T1, младший байт
OCR1AH	$2B ($4B)	Регистр совпадения A таймера/счетчика T1, старший байт
OCR1AL	$2A ($4A)	Регистр совпадения A таймера/счетчика T1, младший байт
OCR1BH	$29 ($49)	Регистр совпадения B таймера/счетчика T1, старший байт
OCR1BL	$28 ($48)	Регистр совпадения B таймера/счетчика T1, младший байт
ICR1H	$27 ($47)	Регистр захвата таймера/счетчика T1, старший байт
ICR1L	$26 ($46)	Регистр захвата таймера/счетчика T1, младший байт
TCCR2	$25 ($45)	Счетный регистр таймера/счетчика T2
TCNT2	$24 ($44)	Регистр совпадения таймера/счетчика T2

Продолжение таблицы 2.20

Название	Адрес	Функция
OCR2	$23 ($43)	Регистр совпадения таймера/счетчика T2
OCDR	$22 ($42)	Регистр внутрисхемной отладки
WDTCR	$21 ($41)	Регистр управления сторожевым таймером
SFIOR	$20 ($40)	Регистр специальных функций
EEARH	$1F ($3F)	Регистр адреса EEPROM, старший байт
EEARL	$1E ($3E)	Регистр адреса EEPROM, младший байт
EEDR	$1D ($3D)	Регистр данных EEPROM
EECR	$1C ($3C)	Регистр управления EEPROM
PORTA	$1B ($3B)	Регистр данных порта A
DDRA	$1A ($3A)	Регистр направления данных порта A
PINA	$19 ($39)	Выводы порта A
PORTB	$18 ($38)	Регистр данных порта B
DDRB	$17 ($37)	Регистр направления данных порта B
PINB	$16 ($36)	Выводы порта B
PORTC	$15 ($35)	Регистр данных порта C
DDRC	$14 ($34)	Регистр направления данных порта C
PINC	$13 ($33)	Выводы порта C
PORTD	$12 ($32)	Регистр данных порта D
DDRD	$11 ($31)	Регистр направления данных порта D
PIND	$10 ($30)	Выводы порта D
SPDR	$0F ($2F)	Регистр данных SPI
SPSR	$0E ($2E)	Регистр состояния SPI
SPCR	$0D ($2D)	Регистр управления SPI
UDR0	$0C ($2C)	Регистр данных USART0
UCSR0A	$0B ($2B)	Регистр управления и состояния A USART0
UCSR0B	$0A ($2A)	Регистр управления и состояния B USART0
UBRR0L	$09 ($29)	Регистр скорости передачи USART0, младший байт
ACSR	$08 ($28)	Регистр управления и состояния аналогового компаратора
ADMUX	$07 ($27)	Регистр управления мультиплексором АЦП
ADCSRA	$06 ($26)	Регистр управления и состояния АЦП
ADCH	$05 ($25)	Регистр данных АЦП, старший байт
ADCL	$04 ($24)	Регистр данных АЦП, младший байт
PORTE	$03 ($23)	Регистр данных порта E
DDRE	$02 ($22)	Регистр направления данных порта E
PINE	$01 ($21)	Выводы порта E
PINF	$00 ($20)	Выводы порта F

К РВВ, 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 пространством ввода/вывода. В системе команд имеется также четыре команды поразрядного доступа, использующие в качестве операндов регистры ввода/вывода: команды установки/сброса отдельного бита (SBI и CBI) и команды проверки состояния отдельного бита (SBIS и SBIC). К сожалению, эти команды могут обращаться только к 1-й половине основных регистров 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 соответствующих 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 всех РВВ есть один регистр, используемый наиболее часто в процессе выполнения программ. Этим регистром является регистр состояния SREG. Он располагается по адресу $3F ($5F) и содержит набор флагов, показывающих текущее состояние микроконтроллера. Большинство флагов автоматически устанавливаются в «1» или сбрасываются в «0» при наступлении определенных событий (в соответствии с результатом выполнения команд). Все разряды этого регистра доступны как для чтения, так и для записи; после сброса микроконтроллера все разряды регистра сбрасываются в «0». Формат этого регистра показан на **Рис. 2.18**, а его описание приведено в **Табл. 2.21**.

Остальные РВВ будут подробно описаны в соответствующих разделах книги.

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

ATMEL-0109

Рис. 2.18. Формат регистра состояния SREG

Таблица 2.21. **Разряды регистра состояния SREG**

Разряд	Название	Описание
7	I	**Общее разрешение прерываний**. Для разрешения прерываний этот флаг должен быть установлен в «1». Разрешение/запрещение отдельных прерываний производится установкой или сбросом соответствующих разрядов регистров масок прерываний (регистра управления прерываниями). Если флаг сброшен, то прерывания запрещены независимо от состояния разрядов этих регистров. Флаг сбрасывается аппаратно после входа в прерывание и восстанавливается командой RETI для разрешения обработки следующих прерываний
6	T	**Хранение копируемого бита**. Этот разряд регистра используется в качестве источника или приемника команд копирования битов BLD (Bit LoaD) и BST (Bit STore). Заданный разряд любого РОН может быть скопирован в этот разряд командой BST или установлен в соответствии с содержимым данного разряда командой BLD
5	H	**Флаг половинного переноса.** Этот флаг устанавливается в «1», если произошел перенос из младшей половины байта (из 3-го разряда в 4-й) или заем из старшей половины байта при выполнении некоторых 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). Более подробно — см. описание системы команд
2	N	**Флаг отрицательного значения.** Этот флаг устанавливается в «1», если старший (7-й) разряд результата операции равен «1». В противном случае флаг равен «0»
1	Z	**Флаг нуля**. Этот флаг устанавливается в «1», если результат выполнения операции равен нулю
0	C	**Флаг переноса.** Этот флаг устанавливается в «1», если в результате выполнения операции произошел выход за границы байта

9.2.2.4. Использование внешнего ОЗУ

Микроконтроллеры ATmega8515x, ATmega161x, ATmega162x, ATmega64x и ATmega128x имеют возможность подключения внешнего статического ОЗУ объемом до 64 Кбайт. Для разрешения работы с внешним ОЗУ необходимо установить в «1» разряд SRE регистра MCUCR (см. ниже).

Выводы микроконтроллеров, используемые для подключения внешнего ОЗУ, сведены в **Табл. 2.22**. Как видно из таблицы, во всех моделях эти выводы являются линиями портов ввода/вывода общего назначения. При разрешении работы с внешним ОЗУ режим работы этих выводов определяется не содержимым регистра направления передачи данных, а самим микроконтроллером.

Таблица 2.22. **Выводы, используемые для подключения внешнего ОЗУ**

Название	ATmega8515x	ATmega161x	ATmega162x	ATmega64x	ATmega128x	Описание
AD0...AD7	PA0...PA7 (PORTA)					Мультиплексированная ША/ШД
A8...A15	PC0...PC7 (PORTC)					Старший байт ША
ALE	PE1			PG2		Строб адреса
$\overline{RD}$	PD7			PG1		Строб записи
$\overline{WR}$	PD6			PG0		Строб чтения

Если работа с внешним ОЗУ разрешена, то при обращении по адресу, находящемуся вне границы внутреннего ОЗУ, автоматически происходит обращение к внешнему ОЗУ. После формирования на выводах порта A младшего байта адреса вывод ALE меняет свое состояние с лог. 1 на лог. 0 и остается в этом состоянии в течение всего цикла чтения/записи. Обращение к внутреннему ОЗУ при разрешенной работе с внешним ОЗУ также может привести к некоторой активности на выводах портов A и C, однако это не влияет на работу схемы, поскольку сигналы стробов чтения ($\overline{RD}$) и записи ($\overline{WR}$) при этом находятся в неактивном состоянии.

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» разряд XMBK регистра SFIOR (модели ATmega8515x и ATmega162x) или XMCRB (модели ATmega64x и ATmega128x). При установленном разряде на выводах порта A всегда сохраняется последнее выведенное значение.

Подключение внешнего ОЗУ к микроконтроллеру показано на **Рис. 2.19**. Как видно из рисунка, для этого дополнительно потребуется регистр-защелка. В качестве такой защелки, как правило, используют микросхему типа 74x573 (например, SN74AHC573) или аналогичную, в которой защелкивание данных происходит по низкому уровню управляющего сигнала.

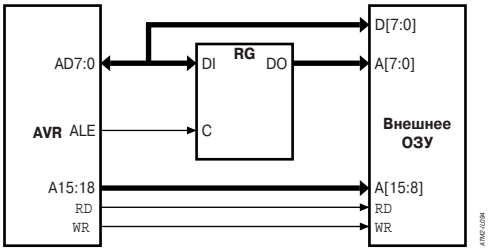


Рис. 2.19. Подключение внешнего ОЗУ к микроконтроллеру

Все рассматриваемые в этом параграфе микроконтроллеры имеют следующие возможности по работе с внешней памятью:

- управление длительностью цикла обращения к внешней памяти;
- разделение 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 токопотребления.

Для управления описанными возможностями во всех микроконтроллерах, кроме ATmega161x, используются по три регистра (в ATmega161x — два), которые перечислены в **Табл. 2.23**.

Таблица 2.23. **Регистры для управления внешней памятью**

Название	Описание	Адрес	Модель
MCUCR	Регистр управления микроконтроллером	$35 ($55)	Все модели
XMCRA	Регистр управления внешней памятью A	($6D)	ATmega64x ATmega128x
EMCUCR	Дополнительный регистр управления микроконтроллером	$36 ($56)	ATmega8515x ATmega161x ATmega162x
XMCRB	Регистр управления внешней памятью B	($6C)	ATmega64x ATmega128x
SFIOR	Регистр специальных функций	$30 ($50)	ATmega8515x ATmega162x

Формат регистра MCUCR для всех моделей приведен на **Рис. 2.20**. Для работы с внешней памятью в нем используются только два разряда (**Табл. 2.24**).

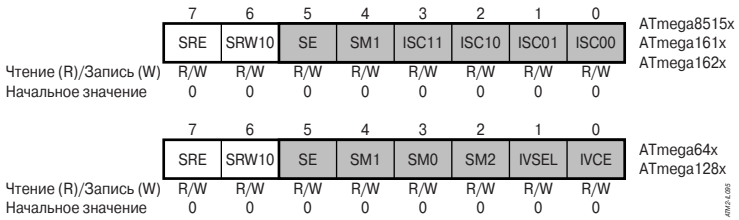


Рис. 2.20. Формат регистра MCUCR

Таблица 2.24. **Разряды регистра MCUCR при управлении внешней памятью**

Разряд	Название	Описание
7	SRE	**Разрешение работы с внешней памятью.** Установка этого разряда в «1» разрешает работу с внешней памятью. Установки регистров направления передачи данных для соответствующих выводов **(Табл. 2.22)** при этом игнорируются. При сброшенном разряде SRE обращение к внешней памяти запрещено, а выводы используются как линии ввода/вывода общего назначения
6	SRW10	**Выбор числа тактов ожидания (верхний сектор).** Этот разряд является младшим разрядом селектора длительности обращения ко второму (верхнему) сектору памяти

Формат регистров EMCUCR и XMCRA приведен на **Рис. 2.21**. В этих регистрах для работы с внешней памятью используются только 6 разрядов. Однако в регистре EMCUCR два оставшихся разряда используются для других целей, а в регистре XMCRA они не используются вообще. Описание используемых для управления внешней памятью разрядов этих регистров приведено в **Табл. 2.25**.

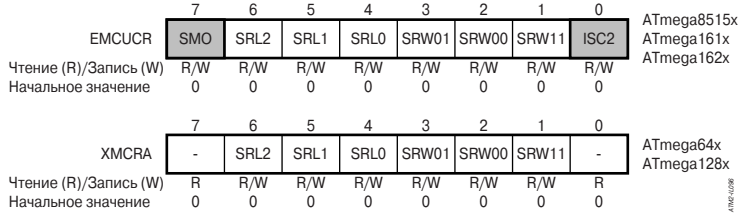


Рис. 2.21. Формат регистров EMCUCR и XMCRA

Таблица 2.25. **Раряды регистров XMCRA и EMCUCR**

Разряд	Название	Описание
6	SRL2	**Задание границ секторов.** Значение этих разрядов определяет положение границы между двумя секторами внешней памяти
5	SRL1	
4	SRL0	
3	SRW01	**Выбор числа тактов ожидания (нижний сектор).** Эти разряды являются селектором длительности обращения к первому (нижнему) сектору внешней памяти
2	SRW00	
1	SRW11	**Выбор числа тактов ожидания (верхний сектор).** Этот разряд является старшим разрядом селектора длительности обращения ко второму (верхнему) сектору памяти (младший разряд – в регистре MCUCR)

Формат регистров SFIOR и XMCRB приведен на **Рис. 2.22**. В этих регистрах для работы с внешней памятью используются только 4 разряда. Как и в регистре EMCUCR, в регистре SFIOR остальные разряды используются для других целей. Описание используемых для управления внешней памятью разрядов регистров XMCRB и SFIOR приведено в **Табл. 2.26**.

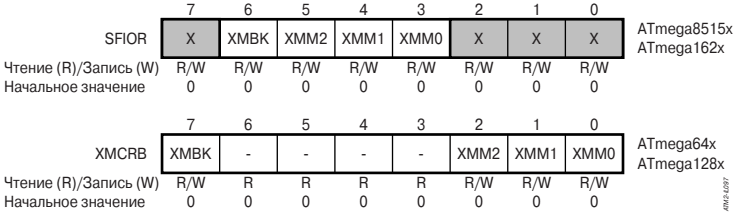


Рис. 2.22. Формат регистров XMCRB и SFIOR

Таблица 2.26. **Разряды регистров SFIOR и XMCRB**

Разряд		Название	Описание
XMCRB	SFIOR		
7	6	XMBK	**Разрешение удержания значений на шине данных.** Если этот разряд установлен в «1», то на выводах AD7...AD0 всегда удерживается последнее выведенное значение (даже если микроконтроллер должен перевести их в третье состояние). При сброшенном разряде эта функция отключена. **Установка разряда XMBK действует, даже если работа с внешней памятью запрещена!**
2	5	XMM2	**Маска старшего байта шины адреса.** Содержимое этих разрядов определяет количество выводов порта C, задействованных под шину адреса. Незадействованные выводы могут использоваться как линии ввода/вывода общего назначения
1	4	XMM1	
0	3	XMM0	

Микроконтроллеры семейства Mega позволяют использовать микросхемы внешнего ОЗУ с различным временем доступа. Подстройка под

различные микросхемы осуществляется изменением длительности цикла обращения к внешней памяти. Более того, имеется возможность разбить все адресное пространство внешнего ОЗУ на два сектора, для каждого из которых может быть задана своя длительность цикла обращения. В общем виде конфигурация внешнего ОЗУ, подключенного к микроконтроллеру, показана на **Рис. 2.23**.

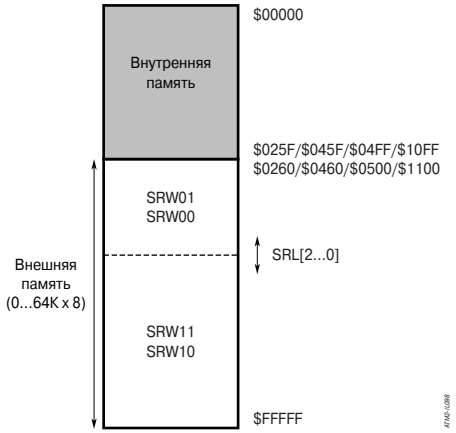


Рис. 2.23. Конфигурация внешнего ОЗУ

Положение границы между секторами определяется содержимым разрядов SRL2...SRL0 согласно **Табл. 2.27**. По умолчанию эти разряды сброшены в «0», т. е. вся область памяти состоит из одного сектора.

Таблица 2.27. **Определение секторов внешней памяти**

SRL2	SRL1	SRL0	Нижняя граница 1-го сектора	Верхняя граница 2-го сектора
0	0	0	—	$0260 — ATmega8515x $0460 — ATmega161x* $0500 — ATmega162x $1100 — ATmega64x/128x
0	0	1	$1FFF	$2000
0	1	0	$3FFF	$4000
0	1	1	$5FFF	$6000
1	0	0	$7FFF	$8000
1	0	1	$9FFF	$A000
1	1	0	$AFFF	$C000
1	1	1	$DFFF	$E000
* Также ATmega162x в режиме совместимости с ATmega161x.				

Изменение длительности цикла обращения к внешней памяти осуществляется заданием дополнительных тактов ожидания с помощью разрядов SRW01:SRW00 для первого сектора и SRW11:SRW10 для второго (см. **Табл. 2.28**).

Таблица 2.28. **Определение числа тактов ожидания**

SRW*n*1*	SRW*n*0*	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
* *n* = 0 (нижний) или 1 (верхний) сектор.		

Временные диаграммы обращения к внешнему ОЗУ при различных установках разрядов SRW*n*1:SRW*n*0 приведены на **Рис. 2.24**.

Как уже было сказано, при работе с внешним ОЗУ используется 16-разрядная шина адреса, позволяющая адресовать 64К адресов. Для многих приложений не требуется такого большого объема внешней памяти. Специально для таких случаев почти все микроконтроллеры, за исключением моделей ATmega161x, позволяют уменьшить разрядность шины адреса. Число выводов порта C, задействованных под шину адреса, определяется содержимым разрядов XMM2...XMM0 в соответствии с **Табл. 2.29**. Незадействованные выводы могут использоваться как линии ввода/вывода общего назначения.

Таблица 2.29. **Задание разрядности старшего байта шины адреса**

XMM2	XMM1	XMM0	Кол-во разрядов, задействованных под шину адреса	Высвобожденные контакты ввода/вывода
0	0	0	8	Нет
0	0	1	7	PC7
0	1	0	6	PC7, PC6
0	1	1	5	PC7...PC5
1	0	0	4	PC7...PC4
1	0	1	3	PC7...PC3
1	1	0	2	PC7...PC2
1	1	1	Старший байт адреса не используется	Весь порт C

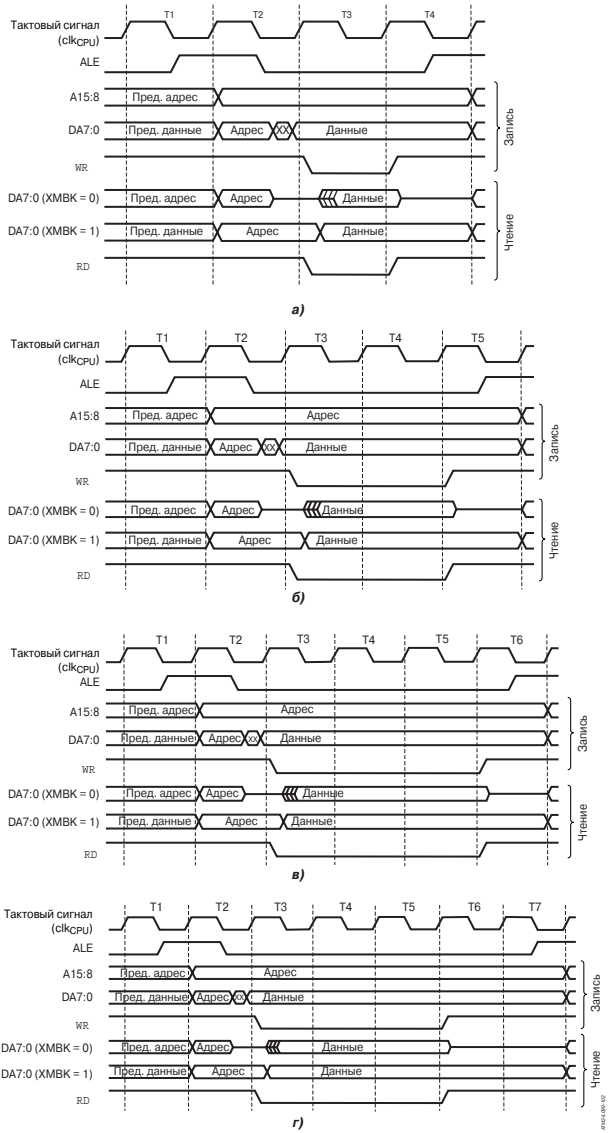


Рис. 2.24. Временные диаграммы обращения к внешнему ОЗУ:
а — SRW*n*1:SRW*n*0 = «00»; *б* — SRW*n*1:SRW*n*0 = «01»;
в — SRW*n*1:SRW*n*0 = «10»; *г* — SRW*n*1:SRW*n*0 = «11»

С помощью разрядов XMM2...XMM0 можно также обеспечить полное использование емкости микросхем внешнего ОЗУ. По умолчанию, как показано на **Рис. 2.23**, младшие адреса внешней памяти отображаются на адресное пространство внутреннего ОЗУ и часть емкости внешнего ОЗУ остается незадействованным. Полностью использовать емкость внешнего ОЗУ можно, программно управляя старшими разрядами адреса. Для выполнения этого «трюка» порт C должен быть настроен на вывод, а в защелке порта должно быть записано «$00». При маскировании старших разрядов адреса на соответствующие выводы порта C будут выставлены «0» и произойдет обращение к младшим адресам внешнего ОЗУ начиная с адреса $0000. Ниже приведены два примера программной реализации описанных действий для микроконтроллера ATmega128.

Пример на ассемблере

```
;Константа OFFSET определена как $2000 для гарантированного
;обращения к внешней памяти
ldi      r16,$FF
out      DDRC,r16             ;Все выводы порта C выходы
ldi      r16,$00
out      PORTC,r16            ;Записали в защелку порта $00
ldi      r16,(1<<XMM1)|(1<<XMM0)
out      XMCRB,r16            ;Высвободили разряды PC7…PC5
ldi      r16,$AA
sts      $0001+OFFSET,r16     ;Записали $AA по адресу $0001
                              ;внешней памяти
ldi      r16,(0<<XMM1)|(0<<XMM0)
out      XMCRB, r16           ;Переназначили разряды PC7…PC5
ldi      r16,0 × 55
sts      0x0001 + OFFSET,r16  ;Записали $55 по адресу (OFFSET + 1)
                              ;внешней памяти
```

Пример на C

```
#define OFFSET 0 x 2000
void XRAM_example(void)
{
unsigned char *p = (unsigned char *) (OFFSET + 1);
DDRC = 0 × FF;
PORTC = 0 × 00;
XMCRB = (1<<XMM1) | (1<<XMM0);
*p = 0xaa;
XMCRB = 0 × 00;
*p = 0 × 55;
}
```

Необходимо помнить, что по сравнению с обращением к внутреннему ОЗУ обращение к внешнему ОЗУ требует на 1, 2, 3 или 4 (зависит от режима обращения) машинных цикла больше для каждого байта, обрабатываемого командой. Таким образом, время выполнения команд передачи данных (LD, ST, LDS, STS, PUSH и POP) увеличивается на 1 (2, 3, 4) цикла. Если стек расположен во внешнем ОЗУ, то время перехода к обработке прерываний, вызова и возврата из подпрограмм увеличивается на 3 (5, 7, 9) машинных цикла. Это связано с тем, что во время выполнения указанных операций происходит сохранение или восстановление 16-разрядного счетчика команд и, кроме того, при обращении к внешней памяти не используется конвейеризация.

9.2.2.5. 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 дополнительные действия выполняются над индексным регистром (при косвенной адресации).

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

Прямая адресация

При прямой адресации адреса операндов содержатся непосредственно в слове команды. В соответствии со структурой памяти данных существуют следующие разновидности прямой адресации: прямая адресация одного РОН, прямая адресация двух РОН, прямая адресация РВВ, прямая адресация ОЗУ.

Прямая адресация одного регистра общего назначения

Этот способ адресации используется в командах, оперирующих с одним из регистров общего назначения. При этом адрес регистра-операнда (его номер) содержится в пяти разрядах слова команды (**Рис. 2.25**).

Примером команд, использующих этот способ адресации, являются команды работы со стеком (PUSH, POP), команды инкремента (INC), декремента (DEC), а также некоторые команды арифметических операций.

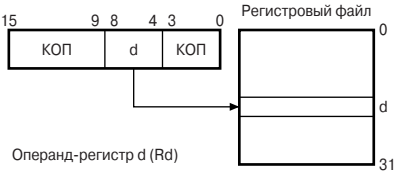


Рис. 2.25. Прямая адресация одного регистра общего назначения

Прямая адресация двух регистров общего назначения

Этот способ адресации используется в командах, оперирующих одновременно с двумя регистрами общего назначения. При этом адрес регистра-источника содержится в разрядах 9, 3...0 (5 разрядов), а адрес регистра-приемника в разрядах 8...4 (5 разрядов) слова команды (**Рис. 2.26**). Положение разрядов r и d на рисунке показано условно.

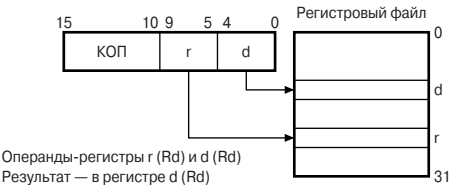


Рис. 2.26. Прямая адресация двух регистров общего назначения

К командам, использующим этот способ адресации, относятся команда 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 используют 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 действительности 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, расположенного в основном пространстве ввода/вывода, и регистровым файлом — IN и OUT. В этом случае адрес регистра ввода/вывода содержится в разрядах 10, 9, 3...0 (6 разрядов), а адрес РОН — в разрядах 8...4 (5 разрядов) слова команды (**Рис. 2.27**). Положение разрядов r/d и P на рисунке показано условно.

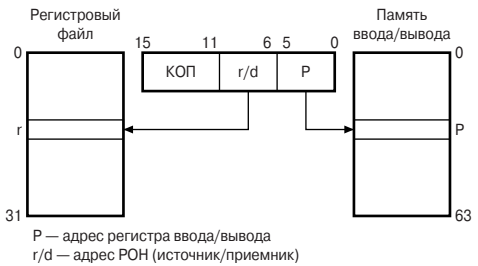


Рис. 2.27. Прямая адресация регистра ввода/вывода

Прямая адресация ОЗУ

Данный способ используется для обращения ко всему адресному пространству 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 разряда). В первом слове содержится код операции и адрес регистра общего назначения (в разрядах с 8-го по 4-й). Во втором слове находится адрес ячейки памяти, к которой происходит обращение (**Рис. 2.28**).

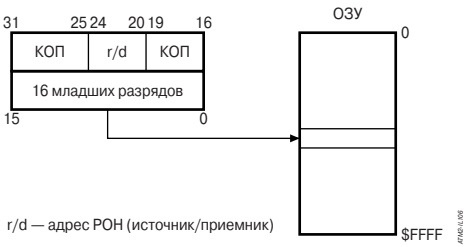


Рис. 2.28. Прямая адресация ОЗУ

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 косвенная 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 регистре (**Рис. 2.29**). Никаких действий с содержимым индексного регистра при этом не производится.

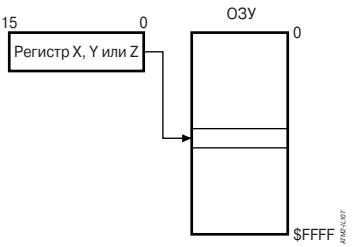


Рис. 2.29. Простая косвенная адресация

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). Адрес регистра общего назначения содержится в разрядах 8...4 слова команды.

Относительная косвенная адресация

При использовании команд относительной косвенной адресации адрес ячейки памяти, к которой производится обращение, получается суммированием содержимого индексного регистра (Y или Z) и константой, задаваемой в команде. Другими словами, производится обращение по адресу, указанному в команде, относительно адреса, находящегося в индексном регистре. Иллюстрация данного способа адресации приведена на **Рис. 2.30**. Положение разрядов n и q на рисунке показано условно.

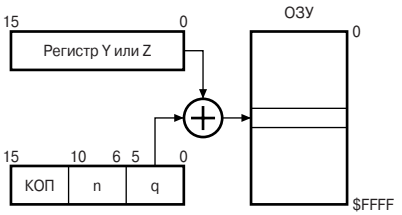


Рис. 2.30. Относительная косвенная адресация

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 разрядах 8...4 слова команды, а величина смещения — в разрядах 13, 11, 10, 2...0. Поскольку под значение смещения отводится только 6 разрядов, 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. 2.31**).

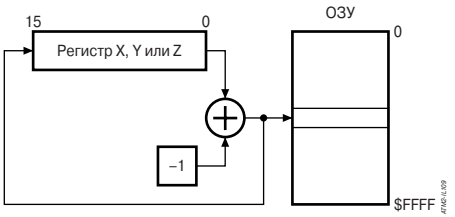


Рис. 2.31. Косвенная адресация с преддекрементом

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 разрядах 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. 2.32**).

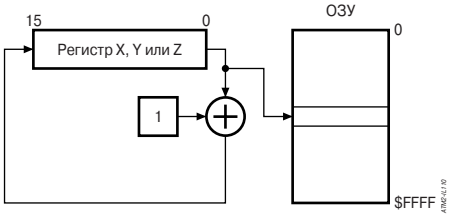


Рис. 2.32. Косвенная адресация с постинкрементом

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 разрядах 8...4 слова команды.

9.2.3. Энергонезависимая память данных (EEPROM)

Все микроконтроллеры семейства Mega имеют в своем составе энергонезависимую память (EEPROM-память). Объем этой памяти колеблется от 512 байт в моделях ATmega8x до 4 Кбайт в старших моделях. EEPROM-память расположена в своем адресном пространстве и так же, как и ОЗУ, организована линейно. Для работы с EEPROM-памятью используются три регистра ввода/вывода: регистр адреса, регистр данных и регистр управления.

Регистр адреса

Регистр адреса EEPROM-памяти EEAR (EEPROM Address Register) физически размещается в двух РВВ EEARH:EEARL, расположенных по адресам $1F ($3F) и $1E ($3E) соответственно. В этот регистр загружается адрес ячейки, к которой будет производиться обращение. Регистр адреса доступен как для записи, так и для чтения. При этом в регистре EEARH задействуются только младшие разряды (количество задействованных разрядов зависит от объема EEPROM-памяти). Незадействованные разряды регистра EEARH доступны только для чтения и содержат «0».

Регистр данных

Регистр данных EEPROM-памяти EEDR (EEPROM Data Register) расположен по адресу $1D ($3D). При записи в этот регистр загружаются дан-

ные, которые должны быть помещены в EEPROM, а при чтении в этот регистр помещаются данные, считанные из EEPROM.

Регистр управления

Регистр управления EEPROM-памяти EECR (EEPROM Control Register) расположен по адресу $1C ($3C). Этот регистр используется для управления доступом к EEPROM-памяти. Формат этого регистра показан на **Рис. 2.33**, а его описание приведено в **Табл. 2.30**.

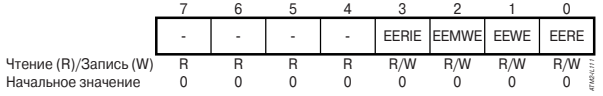


Рис. 2.33. Формат регистра EECR

Таблица 2.30. **Разряды регистра EECR**

Разряд	Название	Описание
7...4	—	Не используются, читаются как «0»
3	EERIE	**Разрешение прерывания от EEPROM.** Этот разряд управляет генерацией прерывания, возникающего при завершении цикла записи в EEPROM. Если этот разряд установлен в «1», прерывания разрешены (если флаг I регистра SREG также установлен в «1»). При сброшенном разряде EEWE (см. далее в таблице) прерывание генерируется постоянно
2	EEMWE	**Управление разрешением записи в EEPROM.** Состояние этого разряда определяет функционирование флага разрешения записи EEWE. Если данный разряд установлен в «1», то при записи в разряд EEWE «1» происходит запись данных в EEPROM. В противном случае установка EEWE в «1» не производит никакого эффекта. После программной установки разряд EEMWE сбрасывается аппаратно через 4 машинных цикла
1	EEWE	**Разрешение записи в EEPROM.** При установке этого разряда в «1» происходит запись данных в EEPROM (если EEMWE равен «1»)
0	EERE	**Разрешение чтения из EEPROM.** После установки этого разряда в «1» выполняется чтение данных из EEPROM. По окончании чтения этот разряд сбрасывается аппаратно

Итак, процедура записи одного байта в EEPROM-память состоит из следующих этапов:

1. Дождаться готовности EEPROM к записи данных (ждать пока не сбросится флаг EEWE регистра EECR).
2. Дождаться завершения записи во FLASH-память программ (ждать пока не сбросится флаг SPMEN регистра SPMCR).
3. Загрузить байт данных в регистр EEDR, а требуемый адрес — в регистр EEAR (при необходимости).
4. Установить в «1» флаг EEMWE регистра EECR.

5. Записать в разряд EEWE регистра EECR лог. «1» в течение 4-х машинных циклов. После установки этого разряда процессор пропускает 2 машинных цикла перед выполнением следующей инструкции.

Второй пункт введен из-за того, что запись в EEPROM-память не может выполняться одновременно с записью в FLASH-память. Поэтому перед выполнением записи в EEPROM-память следует убедиться, что программирование FLASH-памяти завершено. Если в программе отсутствует загрузчик, т. е. микроконтроллер никогда не изменяет содержимое памяти программ, второй шаг может быть пропущен.

Процесс обращения к EEPROM-памяти контролируется внутренним *RC*-генератором (во всех моделях, кроме ATmega161x, для этой цели используется калибруемый *RC*-генератор). Соответственно, длительность цикла записи зависит от частоты этого генератора, напряжения питания, температуры и составляет 2.0...3.8 мс для моделей ATmega161x/163x/323x и 7.5...9.0 мс для остальных моделей. По окончании цикла записи разряд EEWE аппаратно сбрасывается, после чего программа может начать запись следующего байта.

При записи в EEPROM могут возникнуть некоторые проблемы, вызванные прерываниями:

1. При возникновении прерывания между 4-м и 5-м этапами описанной последовательности, запись в EEPROM будет сорвана, т.к. за время обработки прерывания флаг EEMWE сбросится в «0».
2. Если в подпрограмме обработки прерывания, возникшего во время записи в EEPROM-память, также происходит обращение к ней, то будет изменено содержимое регистров адреса и данных EEPROM. В результате первая запись (прерванная) будет сорвана.

Для избежания описанных проблем настоятельно рекомендуется запрещать все прерывания (сбрасывать бит I регистра SREG) на время выполнения пунктов 2...5 описанной выше последовательности.

Ниже приведены два примера реализации функции записи в EEPROM-память (управление прерываниями не производится).

Пример на ассемблере

```
EEPROM_write:
   sbic EECR,EEWE       ;Ждать завершения предыдущей записи
   rjmp EEPROM_write
   out  EEARH,r18
   out  EEARL,r17       ;Занести адрес (r18:r17) в регистр адреса
   out  EEDR,r16        ;Записать данные (r16) в регистр данных
   sbi  EECR,EEMWE      ;Установить флаг EEMWE
   sbi  EECR,EEWE       ;Начать запись в EEPROM
   ret
```

Пример на C

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
while(EECR & (1<<EEWE));  /*Ждать завершения предыдущей записи*/
EEAR = uiAddress;         /*Проинициализировать регистры*/
EEDR = ucData;
EECR |= (1<<EEMWE);       /*Установить флаг EEMWE*/
EECR |= (1<<EEWE);        /*Начать запись в EEPROM*/
}
```

Перед выполнением операции чтения также необходимо проконтролировать состояние флага EEWE. Дело в том, что пока выполняется операция записи в EEPROM-память (флаг EEWE установлен), нельзя выполнять ни чтения EEPROM-памяти, ни изменения регистра адреса.

После загрузки требуемого адреса в регистр EEAR необходимо установить в «1» разряд EERE регистра EECR. Когда запрошенные данные будут помещены в регистр данных EEDR, произойдет аппаратный сброс этого разряда. Однако следить за состоянием разряда EERE для определения момента завершения операции чтения не требуется, т. к. операция чтения из EEPROM всегда выполняется за один машинный цикл. Кроме того, после установки разряда EERE в «1» процессор пропускает 4 машинных цикла перед началом выполнения следующей инструкции.

С учетом сказанного функция чтения из EEPROM может быть реализована следующим образом:

Пример на ассемблере

```
EEPROM_read:
   sbic EECR,EEWE          ;Ждать завершения предыдущей записи
   rjmp EEPROM_read
   out  EEARH, r18
   out  EEARL, r17         ;Занести адрес (r18:r17) в регистр адреса
   sbi  EECR,EERE          ;Начать чтение
   in   r16,EEDR           ;Сохранить данные
   ret
```

Пример на C

```
unsigned char EEPROM_write(unsigned int uiAddress)
{
while(EECR & (1<<EEWE)); /*Ждать завершения предыдущей записи*/
EEAR = uiAddress;        /*Проинициализировать регистр адреса*/
EECR |= (1<<EERE);       /*Выполнить чтение*/
return EEDR;
}
```

Также при использовании EEPROM-памяти необходимо соблюдать некоторые меры предосторожности, чтобы избежать повреждения данных, хранящихся в ней. Подробно эти меры были описаны в подразделе 2.2.3.2 (Часть 1).

9.3. Счетчик команд и выполнение программы

9.3.1. Счетчик команд

Счетчик команд представляет собой регистр, в котором содержится адрес следующей исполняемой команды. Напрямую из программы он недоступен. Размер счетчика команд зависит от объема имеющейся памяти программ и составляет 12 (ATmega8x)...16 (ATmega128x) разрядов.

При нормальном выполнении программы содержимое счетчика команд автоматически увеличивается на 1 или на 2 (в зависимости от выполняемой команды) в каждом машинном цикле. Этот порядок нарушается при выполнении команд перехода, вызова и возврата из подпрограмм, а также при возникновении прерываний.

После включения питания, а также после сброса микроконтроллера в счетчик программ автоматически загружается стартовый адрес $0000 или начальный адрес сектора загрузчика. Как правило, по этому адресу располагается команда безусловного перехода к инициализационной части программы.

При возникновении прерывания в счетчик команд загружается адрес соответствующего вектора прерывания. Если прерывания используются в программе, по адресам векторов прерываний должны размещаться команды перехода к подпрограммам обработки прерываний.

9.3.2. Функционирование конвейера

Как и во всех микроконтроллерах AVR, в микроконтроллерах семейства Mega используется конвейерная обработка команд. Функционирование конвейера показано на **Рис. 2.34**.

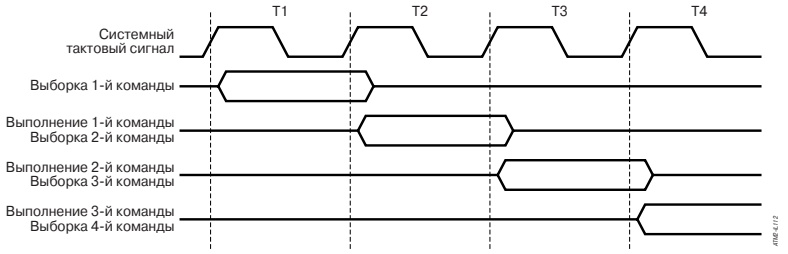


Рис. 2.34. Последовательность выполнения команд в конвейере

Во время первого машинного цикла происходит выборка команды из памяти программ и ее декодирование. Во время второго цикла эта команда выполняется, а параллельно происходит выборка и декодирование второй команды и т. д. В результате фактическое время выполнения каждой команды получается равным одному машинному циклу.

Однако в действительности при отработке ряда команд происходит нарушение нормального функционирования конвейера. Наиболее ярким примером команд, вызывающих подобное нарушение, являются команды условного перехода, а также команды типа Test & Skip (проверка и пропуск следующей команды, если результат проверки положительный). В случае истинности условия, проверяемого командой условного перехода, выполнение программы должно будет продолжиться с некоторого адреса. А поскольку в конвейере уже произошла выборка команды, расположенной за командой перехода, время выполнения команды перехода увеличивается на один машинный цикл, во время которого происходит выборка команды, расположенной по требуемому адресу.

При выполнении команд типа Test & Skip, следующая команда не выполняется в случае истинности проверяемого условия. Однако выборка пропускаемой команды уже произошла. Вследствие того что команда не выполняется, в конвейере образуется «дырка», которая заключается в пропуске одного или двух (в зависимости от пропускаемой команды) машинных циклов. Таким образом команды типа Test & Skip выполняются за один машинный цикл, если результат проверки условия отрицателен, и за два или три цикла, если он положителен.

Очевидно, что в результате выполнения любой команды, изменяющей содержимое счетчика команд (команды перехода, вызова и возврата из подпрограмм), также происходит «разрыв» в работе конвейера, вследствие чего происходит задержка выполнения программы на несколько машинных циклов. Длительность задержки составляет от двух до четырех машинных циклов в зависимости от команды. Для получения более подробной информации обратитесь к описанию команд (Часть 3).

9.3.3. Команды типа «проверка/пропуск» (Test & Skip)

В командах этого типа производится проверка условия, результат которой влияет на выполнение следующей команды. Если условие истинно, следующая команда игнорируется. Например, команда SBRS Rd,b проверяет разряд «b» регистра Rd и игнорирует следующую команду, если этот разряд равен «1». При истинности проверяемого условия в конвейере возникает задержка, связанная с необходимостью загрузки и декодирования новой команды. Длительность задержки зависит от размера пропускаемой команды и составляет от одного до трех машинных циклов.

9.3.4. Команды условного перехода

В этих командах производится проверка условия, результат которой влияет на состояние счетчика команд. Если условие истинно, происходит переход по заданному адресу. Если же условие ложно, выполняется следующая команда.

Команды условного перехода имеют ограничение по области действия. В действительности новое значение счетчика команд получается прибавлением к нему или вычитанием из него некоторого смещения. А поскольку под значение смещения в слове команды отводится всего 7 разрядов, максимальная величина перехода составляет –63...+64 слов.

Так как переход по заданному адресу осуществляется загрузкой нового значения в счетчик команд, то в случае истинности проверяемого условия в конвейере возникает задержка длительностью в один машинный цикл.

9.3.5. Команды безусловного перехода

Все рассматриваемые в книге микроконтроллеры семейства Mega имеют 3 команды безусловного перехода: команду относительного перехода RJMP, команду абсолютного перехода JMP, а также команду косвенного перехода IJMP.

Относительный переход — команда RJMP

При выполнении команды относительного перехода содержимое счетчика команд изменяется прибавлением к нему или вычитанием из него некоторого значения, являющегося операндом команды, как показано на **Рис. 2.35**. Положение разрядов k на рисунке показано условно. Поскольку под значение операнда в слове команды отводится всего 11 разрядов, с помощью этой команды можно переходить только в пределах –2047...+2048 слов.

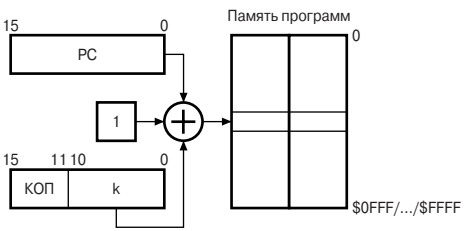


Рис. 2.35. Относительная адресация памяти программ

В программах в качестве операндов этой команды вместо констант используются метки. Ассемблер сам вычисляет величину перехода и подставляет это значение в слово команды.

Поскольку команда относительного перехода изменяет содержимое счетчика команд, она выполняется за 2 машинных цикла.

Абсолютный переход — команда JMP

При выполнении команды абсолютного перехода в счетчик команд просто загружается значение нового адреса, являющееся операндом команды (**Рис. 2.36**). Положение разрядов k на рисунке показано условно. С помощью этой команды можно осуществлять переход в пределах всего адресного пространства имеющихся на сегодняшний день микроконтроллеров AVR.

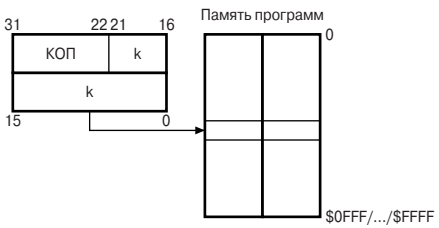


Рис. 2.36. Абсолютная адресация памяти программ

Команда абсолютного перехода выполняется за 3 машинных цикла. Дополнительный (по сравнению с командой относительного перехода) цикл связан с большей длиной кода команды (2 слова).

Косвенный переход — команда IJMP

При выполнении этой команды осуществляется переход по адресу, который находится в индексном регистре Z. Соответственно процесс выполнения команды сводится к загрузке содержимого индексного регистра в счетчик команд (**Рис. 2.37**).

Как и команда абсолютного перехода, данная команда не имеет ограничений по области действия. Действительно, поскольку индексный регистр 16-разрядный, максимально возможная величина перехода составляет 64 Кслов (128 Кбайт) — именно такой объем памяти программ имеют модели ATmega128x.

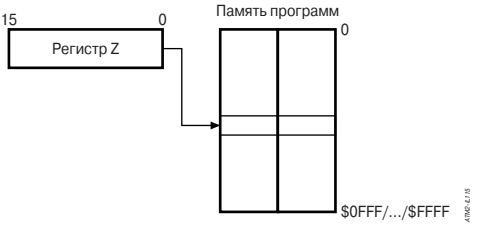


Рис. 2.37. Косвенная адресация памяти программ

Еще раз отметим, что команда не имеет ограничений только для микроконтроллеров семейства, выпущенных на момент написания книги. В будущем вполне возможно появление микроконтроллеров с объемом памяти программ более 128 Кбайт, для которых команда IJMP также будет иметь ограничение по области действия. Забегая вперед, заметим, что для таких устройств в системе команд микроконтроллеров AVR уже предусмотрена команда удлиненного косвенного перехода, позволяющая адресовать 8 Мбайт памяти программ.

Как и команда относительного перехода, команда косвенного перехода выполняется за 2 машинных цикла.

9.3.6. Команды вызова подпрограмм

Как и для реализации безусловных переходов, для вызова подпрограмм в микроконтроллерах семейства Mega имеются три команды: команда относительного вызова RCALL, команда абсолютного вызова CALL, а также команда косвенного вызова ICALL. Если не принимать во внимание некоторые отличия, описанные ниже, эти команды работают так же, как и соответствующие команды безусловного перехода.

Относительный вызов подпрограммы — команда RCALL

Сначала команда RCALL сохраняет в стеке значение счетчика команд. Затем содержимое счетчика команд увеличивается или уменьшается на некоторое значение, являющееся операндом команды (**Рис. 2.35**). Поскольку операнд представляет собой 12-разрядное число, максимальная величина перехода составляет –2047...+2048 слов (±4 Кбайта).

В программах в качестве операндов этой команды, как и в случае команды RJMP, используются метки. Ассемблер сам вычисляет величину перехода и подставляет это значение в слово команды.

Команда относительного вызова подпрограмм выполняется за 3 машинных цикла, два из которых затрачиваются на сохранение в стеке содержимого двухбайтного счетчика команд.

Абсолютный вызов подпрограммы — команда CALL

При выполнении команды после сохранения в стеке текущего значения счетчика команд в последний загружается число, являющееся операндом команды (**Рис. 2.36**). С помощью этой команды можно осуществлять вызов в пределах всего адресного пространства имеющихся на сегодняшний день микроконтроллеров AVR.

Команда абсолютного вызова подпрограмм выполняется за 4 машинных цикла.

Косвенный вызов подпрограммы — команда ICALL

Команда ICALL сначала сохраняет в стеке значение счетчика команд. Затем в счетчик команд загружается содержимое индексного регистра. Максимально возможная величина перехода составляет 64K слов (128 Кбайт), поэтому данная команда также не имеет ограничений по области действия.

Как и команда RCALL, команда косвенного вызова подпрограмм выполняется за 3 машинных цикла.

9.3.7. Команды возврата из подпрограмм

В конце каждой подпрограммы обязательно должна находиться команда возврата из нее. В системе команд микроконтроллеров семейства имеется две таких команды. Для возврата из обычной подпрограммы, вызываемой командой RCALL, используется команда RET. Для возврата из подпрограммы обработки прерывания используется команда RETI.

Обе команды восстанавливают из стека содержимое счетчика команд, сохраненное там перед переходом к подпрограмме. Команда возврата из подпрограммы RETI дополнительно устанавливает в «1» флаг общего разрешения прерываний I регистра SREG, сбрасываемый аппаратно при возникновении прерывания.

На выполнение каждой из команд возврата из подпрограммы требуется 4 машинных цикла.

9.4. Стек

Во всех микроконтроллерах семейства Mega стек реализован программно. Он размещается в памяти данных, и его глубина определяется только размером свободной области памяти программ. В качестве указателя стека во всех моделях используется пара регистров ввода/вывода SPH:SPL, расположенных по адресам $3E ($5E) и $3D ($5D) соответственно. Так как после подачи напряжения питания (или после сброса) в регистрах содержится ну-

левое значение, в самом начале программы указатель стека необходимо проинициализировать, записав в него значение верхнего адреса памяти данных.

При вызове подпрограмм адрес команды, расположенной за командой вызова, сохраняется в стеке. Значение указателя стека при этом уменьшается на 2, т. к. для хранения счетчика команд требуется 2 байта. При возврате из подпрограммы этот адрес извлекается из стека и загружается в счетчик команд. Значение указателя стека соответственно увеличивается на 2. То же происходит и во время прерывания. При генерации прерывания адрес следующей команды сохраняется в стеке, а при возврате из подпрограммы обработки прерывания он восстанавливается из стека.

Во всех моделях микроконтроллеров семейства Mega стек доступен программно. Для работы со стеком в наборе команд имеется две команды: команда занесения в стек (PUSH) и команда извлечения из стека (POP).

Глава 10. Тактирование, режимы пониженного энергопотребления и сброс

10.1. Общие сведения

С микроконтроллерами семейства Mega могут использоваться самые различные источники тактового сигнала. Прежде всего, это встроенный кварцевый генератор с подключаемым внешним резонатором. Также в качестве тактового может использоваться простейший *RC*-генератор — как внутренний (калибруемый), так и с внешней *RC*-цепочкой. Кроме того, в качестве тактового может использоваться внешний сигнал синхронизации.

Все микроконтроллеры семейства Mega имеют несколько (до 6) режимов пониженного энергопотребления, так называемые «спящие» режимы. Каждый из этих режимов позволяет сократить энергопотребление микроконтроллера в периоды его бездействия. Вход в любой из этих режимов выполняется по команде SLEEP. При выходе микроконтроллера из «спящего» режима производится его реинициализация (сброс).

Сброс микроконтроллера происходит не только при его «пробуждении», но и при наступлении ряда других событий. Этими событиями являются: появление на выводе $\overline{\text{RESET}}$ сигнала НИЗКОГО уровня, включение напряжения питания, снижение напряжения питания ниже минимально допустимого уровня, срабатывание сторожевого таймера, а также получение команды сброса по интерфейсу JTAG.

10.2. Тактовый генератор

Упрощенное устройство синхронизации микроконтроллеров семейства Mega представлено на **Рис. 2.38**.

Как видно из рисунка, на основе системного тактового сигнала формируются дополнительные сигналы, используемые для тактирования различных модулей и блоков микроконтроллера:

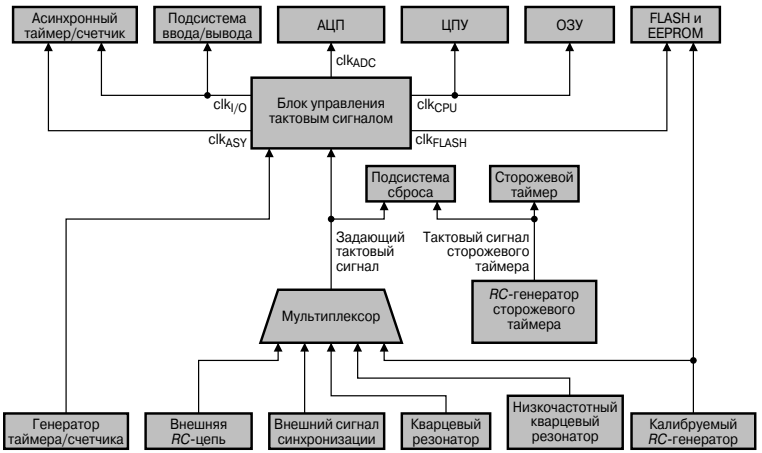


Рис. 2.38. Устройство синхронизации

- clk_{CPU} — тактовый сигнал центрального процессора, используется для тактирования блоков микроконтроллера, отвечающих за работу с ядром микроконтроллера (регистровый файл, память данных и т. п.). При выключении этого сигнала ЦПУ останавливается, все вычисления прекращаются;
- $clk_{I/O}$ — тактовый сигнал подсистемы ввода/вывода, используется большинством периферийных устройств, таких как таймеры/счетчики и интерфейсные модули. Этот сигнал используется также подсистемой внешних прерываний, однако ряд внешних прерываний могут генерироваться и при его отсутствии;
- clk_{FLASH} — тактовый сигнал для управления FLASH-памятью программ. Как правило, этот сигнал формируется одновременно с тактовым сигналом центрального процессора;
- clk_{ASY} — тактовый сигнал асинхронного таймера/счетчика. Тактирование осуществляется непосредственно от внешнего кварцевого резонатора (32768 Гц). Наличие этого сигнала позволяет использовать соответствующий таймер/счетчик в качестве часов реального времени даже при нахождении микроконтроллера в «спящем» режиме;
- clk_{ADC} — тактовый сигнал модуля АЦП. Наличие этого тактового сигнала позволяет осуществлять преобразования при остановленном ЦПУ и подсистеме ввода/вывода. При этом значительно уменьшается уровень помех, генерируемых микроконтроллером, точность преобразования увеличивается.

Тактовый генератор микроконтроллеров семейства Mega может работать с внешним кварцевым резонатором, внешней или внутренней *RC*-цепочкой, а также с внешним сигналом синхронизации. Возможность использования того или иного источника тактового сигнала зависит от модели микроконтроллера (**Табл. 2.31**). Поскольку архитектура микроконтроллеров полностью статическая, минимально допустимая частота ничем не ограничена (вплоть до пошагового режима работы), а максимальная рабочая частота определяется конкретной моделью микроконтроллера.

Таблица 2.31. **Источники тактового сигнала**

Источник тактового сигнала	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega32x	ATmega323x	ATmega64x	ATmega128x
Генератор с внешним резонатором	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
Генеатор с внешней *RC*-цепочкой	♦	♦	♦	—	—	♦	♦	♦	♦	♦
Генератор с внутренней *RC*-цепочкой	♦	♦	♦	—	♦	♦	♦	♦	♦	♦
Внешний сигнал синхронизации	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦

Во всех моделях, за исключением моделей ATmega161x, выбор режима работы осуществляется программированием конфигурационных ячеек (FUSE Bits) CKSEL3...0. Требуемые значения для каждого режима работы приведены в **Табл. 2.32**. Подробно о конфигурационных ячейках будет рассказано в 4-й части книги, посвященной программированию микроконтроллеров.

Таблица 2.32. **Выбор режима работы тактового генератора**

Режим работы	CKSEL3...0		
	ATmega162x	ATmega163x, ATmega323x	Остальные
Кварцевый или керамический резонатор	1111...1000	1111...1010	1111...1010
Низкочастотный кварцевый резонатор	0111...0100	1001...1000	1001
Внешняя *RC*-цепочка	—	0111...0101	1000...0101
Внутренняя *RC*-цепочка*	0010	0100...0010	0100...0001
Внешний сигнал синхронизации	0000	0001...0000	0000
Зарезервировано	0011, 0001	—	—

* Режим по умолчанию.

От значений, занесенных в эти ячейки, зависит также длительность задержки сброса t_{TOUT} (см. 10.4).

10.2.1. Тактовый генератор с внешним резонатором

Резонатор подключается к выводам XTAL1 и XTAL2 микроконтроллеров, как показано на **Рис. 2.39.** Эти выводы являются соответственно входом и выходом инвертирующего усилителя тактового генератора.

Емкости конденсаторов C_1 и C_2, подключаемых между выводами резонатора и общим проводом, зависят от типа резонатора. Для кварцевых резонаторов емкости этих конденсаторов находятся в пределах 12...22 пФ, а для керамических резонаторов должны выбираться согласно рекомендациям производителей резонаторов.

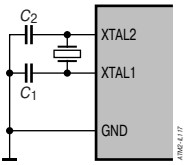


Рис. 2.39. Подключение кварцевого или керамического резонатора

Теперь рассмотрим управление частотой тактового генератора с внешним резонатором (далее, для простоты, кварцевого генератора) в различных моделях.

ATmega8x/8515x/16x/32x/64x/128x

Усилитель тактового генератора этих моделей может работать в одном из двух режимов, определяемом состоянием конфигурационной ячейки CKPOT. Если эта ячейка запрограммирована («0»), размах колебаний на выходе усилителя (вывод XTAL2) практически равен напряжению питания. Данный режим полезен при работе устройства в условиях сильных электромагнитных помех, а также при использовании сигнала тактового генератора для управления внешними устройствами. В последнем случае между выводом и внешней схемой обязательно должен быть буфер.

Если ячейка CKPOT не запрограммирована («1»), размах колебаний на выходе усилителя будет значительно меньше. Соответственно ток потребления микроконтроллера уменьшается, однако при этом сужается и диапазон возможных частот тактового сигнала. Кроме того, в этом режиме сигнал тактового генератора микроконтроллера нельзя использовать для управления внешними устройствами.

Собственно генератор может работать в четырех различных режимах, каждый из которых предназначен для определенного диапазона частот. Эти режимы определяются ячейками CKSEL3...1 и CKPOT (**Табл. 2.33**).

Таблица 2.33. **Режимы работы кварцевого генератора**

CKPOT	CKSEL3...1	Примерный диапазон частот [МГц]
1	101*	0.4...0.9
1	110*	0.9...3.0
1	111	3.0...8.0
0	101, 110, 111	> 1.0
* В этом режиме должен использоваться только керамический резонатор.		

ATmega162x

Генератор этих моделей также может работать в четырех различных режимах, определяемых состоянием ячеек CKSEL3...1. Диапазоны частот, для которых предназначен тот или иной режим, приведены в **Табл. 2.34**.

Таблица 2.34. **Режимы работы кварцевого генератора**

CKSEL3...1	Примерный диапазон частот [МГц]
100*	0.4...0.9
101	0.9...3.0
110	3.0...8.0
111	> 1.0
* В этом режиме должен использоваться только керамический резонатор	

ATmega161x/163x/323x

В этих моделях оптимизация кварцевого генератора под различные частоты отсутствует, а различные установки ячеек CKSEL3...0 в моделях ATmega163x и ATmega323x определяют только длительность задержки сброса t_{TOUT}.

10.2.2. Низкочастотный кварцевый генератор

Режим предназначен для использования кварцевого резонатора на частоту «часового кварца» 32768 Гц. Как и все внешние резонаторы, он подключается к выводам XTAL1 и XTAL2 микроконтроллеров.

В этом режиме большинство микроконтроллеров позволяют подключить между выводами резонатора и общим проводом внутренние конденсаторы. При этом надобность во внешних конденсаторах естественно отпадает. В моделях ATmega8x, ATmega8515x,

ATmega16x, ATmega32x, ATmega64x и ATmega128x конденсаторы емкостью 36 пФ подключаются при записи лог. 0 в конфигурационную ячейку CKPOT. А в моделях ATmega162x конденсаторы подключаются, если в ячейках CKSEL3...0 записано значение «0110» или «0111». Емкость каждого из конденсаторов составляет 10 пФ.

10.2.3. Внешний сигнал синхронизации

Сигнал от внешнего источника подается на вывод XTAL1, как показано на **Рис. 2.40**. Разумеется, этот сигнал должен удовлетворять требованиям микроконтроллера по частоте, скважности и уровням напряжения. Вывод XTAL2 в этом режиме оставляют неподключенным.

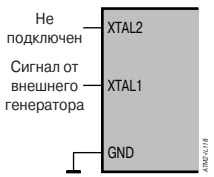


Рис. 2.40. Подключение внешнего источника тактового сигнала

В моделях ATmega8x, ATmega8515x, ATmega16x, ATmega32x, ATmega64x и ATmega128x между выводом XTAL1 и общим проводом можно включить внутренний конденсатор емкостью 36 пФ. Его подключение осуществляется записью «0» в конфигурационную ячейку CKPOT.

10.2.4. Внешняя *RC*-цепочка

При реализации приложений, не требующих высокой временной точности, можно использовать простейший *RC*-генератор. При этом внешняя *RC*-цепочка подключается к выводу XTAL1, как показано на **Рис. 2.41**.

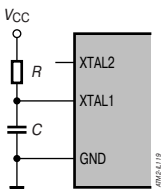


Рис. 2.41. Подключение внешней *RC*-цепочки

Емкость конденсатора цепочки должна быть 22 пФ (min). Сопротивление резистора рекомендуется выбирать из диапазона 3.3...100 кОм.

В моделях ATmega8x, ATmega8515x, ATmega16x, ATmega32x, ATmega64x и ATmega128x частота тактового сигнала определяется формулой $f \approx 1/3RC$.

Внешний конденсатор в этих моделях можно исключить, задействовав внутренний емкостью 36 пФ. Внутренний конденсатор подключается при записи лог. 0 в конфигурационную ячейку CKPOT.

Как и в случае кварцевого генератора, при использовании внешней *RC*-цепочки тактовый генератор может работать в четырех различных режимах, каждый из которых предназначен для определенного диапазона частот.

Эти режимы определяются содержимым ячеек CKSEL3...0 согласно **Табл. 2.35**.

Таблица 2.35. **Режимы работы генератора с внешней *RC*-цепочкой**

CKSEL3...0	Примерный диапазон частот [МГц]
0101	0...0.9
0110	0.9...3.0
0111	3.0...8.0
1000	8.0...12.0

В моделях ATmega163x и ATmega323x частота тактового сигнала определяется формулой $f \approx 2/3RC$.

Значения номиналов R и C для ряда «типовых» частот приведены в **Табл. 2.36**.

Таблица 2.36. **Параметры внешней *RC*-цепочки для «типовых» частот**

R [кОм]	C [пФ]	F [МГц]
100	70	0.1
31.5	20	1
6.5	20	4

10.2.5. Встроенный генератор с внутренней *RC*-цепочкой

Эту опцию имеют почти все микроконтроллеры семейства, за исключением моделей ATmega161x. Использование встроенного *RC*-генератора с внутренней времязадающей *RC*-цепочкой (внутреннего *RC*-генератора) является наиболее экономичным решением, т. к. при этом не требуются никакие внешние компоненты.

Номинальные частоты внутреннего *RC*-генератора для различных моделей приведены в **Табл. 2.37**.

Таблица 2.37. **Номинальные частоты внутреннего *RC*-генератора**

Модель	Частота [МГц]*
ATmega8x	1.0, 2.0, 4.0, 8.0
ATmega8515x	1.0, 2.0, 4.0, 8.0
ATmega16x	1.0, 2.0, 4.0, 8.0
ATmega162x	8.0
ATmega163x	1.0

Продолжение таблицы 2.37

Модель	Частота [МГц]*
ATmega32x	1.0, 2.0, 4.0, 8.0
ATmega323x	1.0
ATmega64x	1.0, 2.0, 4.0, 8.0
ATmega128x	1.0, 2.0, 4.0, 8.0
* При V_{CC} = 5.0 В, T = 25°C	

Как видно из таблицы, внутренний *RC*-генератор большинства микроконтроллеров семейства Mega может работать на нескольких фиксированных частотах. Рабочая частота генератора этих моделей определяется содержимым конфигурационных ячеек CKSEL3...0 согласно **Табл. 2.38**. А в моделях ATmega163x и ATmega323x разные значения этих ячеек, соответствующие режиму внутреннего *RC*-генератора, определяют только длительность задержки сброса t_{TOUT}.

Таблица 2.38. **Режимы работы внутреннего *RC*-генератора**

CKSEL3...0	Частота [МГц]
0001*	1.0
0010	2.0
0011	4.0
0100	8.0
* Режим по умолчанию.	

Следует отметить, что при работе с внутренним *RC*-генератором в конфигурационной ячейке CKPOT (если она имеется в микроконтроллере) должна быть записана лог. 1.

Во всех микроконтроллерах семейства, за исключением моделей ATmega161x, предусмотрена возможность подстройки частоты внутреннего генератора (калибровка). Для этой цели используется регистр OSCCAL, расположение которого в пространстве ввода/вывода для разных моделей приведено в **Табл. 2.39**.

Таблица 2.39. **Адреса регистра OSCCAL**

Модель	Адрес регистра OSCCAL
ATmega8x	$31 ($51)
ATmega8515x	$04 ($24)
ATmega16x	$31 ($51)*
ATmega162x	$04 ($24)*
ATmega163x	$31 ($51)

Продолжение таблицы 2.39

Модель	Адрес регистра OSCCAL
ATmega32x	$31 ($51)*
ATmega323x	$31 ($51)*
ATmega64x	($6F)
ATmega128x	($6F)

* Регистр OSCCAL доступен из программы по указанному адресу только в том случае, если в конфигурационной ячейке OCDEN записана лог. 1.

Чем больше значение, записанное в регистре OSCCAL, тем больше частота генератора. Диапазон возможного изменения частоты *RC*-генератора в зависимости от содержимого этого регистра указан в **Табл. 2.40**.

Таблица 2.40. **Влияние содержимого регистра OSCCAL на частоту внутреннего *RC*-генератора**

Содержимое регистра OSCCAL	f_{MIN} (в процентах от номинальной)	f_{MAX} (в процентах от номинальной)
$00	50	100
$7F	75	150
$FF	100	200

Значение, необходимое для подстройки генератора на номинальную частоту (с точностью ±1%) записывается при изготовлении микроконтроллера в специальные калибровочные ячейки, количество которых равно числу номинальных частот внутреннего *RC*-генератора данной модели. Содержимое этих ячеек доступно только в режиме программирования микроконтроллеров. В микроконтроллерах ATmega163x и ATmega323x программатор должен считать содержимое калибровочной ячейки и записать его в заранее определенное место FLASH-памяти программ (как правило, в последнюю ячейку). А в самом начале программы необходимо прочитать содержимое по этому адресу и загрузить его в регистр OSCCAL.

В остальных моделях загрузка калибровочной константы для частоты 1 МГц осуществляется аппаратно при каждом включении питания. Если же требуется другая частота внутреннего *RC*-генератора, соответствующее значение должно загружаться в регистр OSCCAL программным путем, как описано выше.

Следует помнить, что внутренний генератор предназначен для работы на номинальных частотах. Поэтому подстройка на другие частоты хотя и возможна, но не гарантируется. Более того, внутренний *RC*-генератор определяет временные параметры доступа к FLASH- и EEPROM-памяти, поэтому увеличение частоты генератора более чем на 10% может привести к невозможности записи в эти области памяти.

10.2.6. Управление тактовой частотой

Ряд моделей семейства, а именно ATmega162x, ATmega64x и ATmega128x имеют возможность программного уменьшения частоты сигнала, поступающего от тактового генератора. Понятно, что одновременно с уменьшением тактовой частоты уменьшаются частоты сигналов clk_{CPU}, $clk_{I/O}$, clk_{FLASH}, clk_{ADC}, т. е. замедляется работа всех периферийных устройств микроконтроллера. Если асинхронный таймер/счетчик работает в синхронном режиме, то соответствующим образом изменяется и частота сигнала clk_{ASY}.

В моделях ATmega162x для управления предделителем тактового сигнала предназначен регистр CLKPR, расположенный по адресу ($61) в пространстве дополнительных регистров ввода/вывода. Формат этого регистра приведен на **Рис. 2.42**.

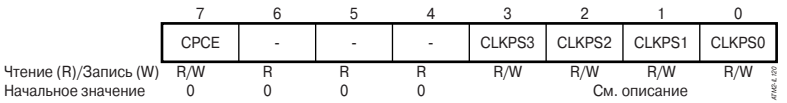


Рис. 2.42. Формат регистра CLKPR

Старший разряд (CPCE) служит для разрешения изменения частоты тактового сигнала, а разряды CLKPS3...CLKPS0 задают коэффициент деления предделителя (**Табл. 2.41**).

Таблица 2.41. **Выбор коэффициента деления предделителя частоты тактового сигнала**

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Коэффициент деления
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16
0	1	0	1	32
0	1	1	0	64
0	1	1	1	128
1	0	0	0	256
1	0	0	1	Зарезервировано
...				
1	1	1	1	

Для изменения содержимого разрядов CLKPS3...0 следует записать в разряд CPCE лог. 1, а в разряды CLKPS3...0 — лог. 0. Затем в течение следующих четырех машинных циклов занести требуемое значение в разряды CLKPS3...0, при этом разряд CPCE будет сброшен в «0».

В противном случае разряд CPCE буден сброшен аппаратно по истечении четырех машинных циклов, запрещая дальнейшее изменение разрядов CLKPS3...0.

Начальное состояние разрядов CLKPS3...0 определяется конфигурационной ячейкой CKDIV16. Если она не запрограммирована («1»), при запуске микроконтроллера в разрядах CLKPS3...0 содержится значение «0000». Если же ячейка CKDIV16 запрограммирована («0»), стартовым значением разрядов CLKPS3...0 является «0100» (коэффициент деления — 16).

В микроконтроллерах ATmega64x и ATmega128x управление тактовой частотой осуществляется немного иначе. В этих моделях для управления предделителем тактового сигнала предназначен регистр ввода/вывода XDIV, расположенный по адресу $3C ($5C). Формат этого регистра приведен на **Рис. 2.43**.

	7	6	5	4	3	2	1	0
	XDIVEN	XDIV6	XDIV5	XDIV4	XDIV3	XDIV2	XDIV1	XDIV0
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

ATMEGA121

Рис.2.43. Формат регистра XDIV

Старший разряд регистра (XDIVEN) служит для включения/выключения предделителя тактового сигнала, а остальные разряды (XDIV6...0) определяют тактовую частоту микроконтроллера. Если обозначить содержимое разрядов XDIV6...0 как *d*, зависимость тактовой частоты от состояния этих разрядов будет определяться выражением F_{CLK} = Частота генератора/(129 – d).

Изменение разрядов XDIV6...0 возможно только при сброшенном разряде XDIVEN. При установке его в «1» тактовая частота микроконтроллера будет определяться выражением, приведенным выше. При сброшенном в «0» разряде XDIVEN содержимое разрядов XDIV6...0 игнорируется.

10.3. Режимы пониженного энергопотребления

Различные модели микроконтроллеров семейства поддерживают 3...6 режимов пониженного энергопотребления (**Табл. 2.42**). Режимы отличаются числом периферийных устройств микроконтроллера, функционирующих в «спящем» режиме и степенью уменьшения энергопотребления.

В зависимости от модели для управления «спящим» режимом используется различное число регистров ввода/вывода, которые сведены в **Табл. 2.43**. Форматы этих регистров приведены на **Рис. 2.44...2.46** (разряды, которые в данном случае не используются, выделены серым цветом).

Таблица 2.42. **Режимы пониженного энергопотребления**

Режим пониженного энергопотребления	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega32x	ATmega323x	ATmega64x	ATmega128x
Idle	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
ADC Noise Reduction	♦	—	♦	—	—	♦	♦	♦	♦	♦
Power Down	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
Power Save	♦	—	♦	♦	♦	♦	♦	♦	♦	♦
Standby	♦	♦	♦	—	♦	—	♦	♦	♦	♦
Extended Standby	—	—	♦	—	♦	—	♦	♦	♦	♦

Таблица 2.43. **Регистры для управления «спящим» режимом**

Название	Описание	Адрес	Рис.	Модель
MCUCR	Регистр управления микроконтроллером	$35 ($55)	2.43	ATmega8x ATmega16x ATmega163x ATmega32x ATmega323x ATmega64x ATmega128x
MCUCR	Регистр управления микроконтроллером	$35 ($55)	2.43	ATmega8515x ATmega162x
MCUCSR	Регистр управления и состояния микроконтроллера	$34 ($54)	2.44	
MCUCR	Регистр управления микроконтроллером	$35 ($55)	2.43	ATmega161x
EMCUCR	Дополнительный регистр управления микроконтроллером	$36 ($56)	2.45	

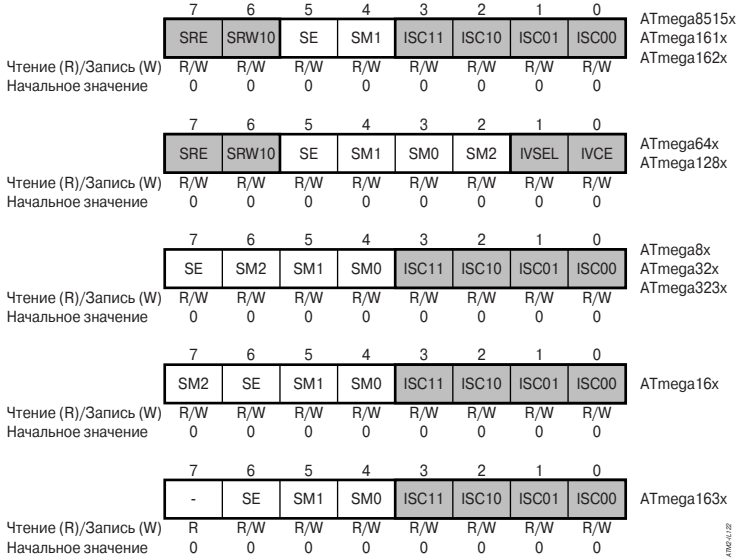


Рис. 2.44. Формат регистра MCUCR

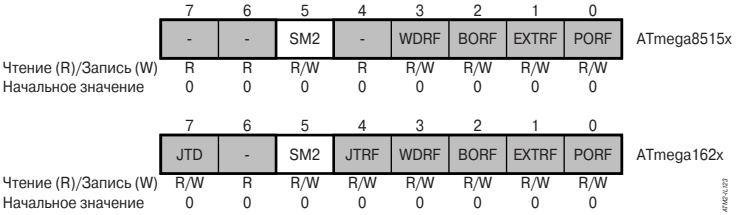


Рис. 2.45. Формат регистра MCUCSR моделей ATmega8515x и ATmega162x

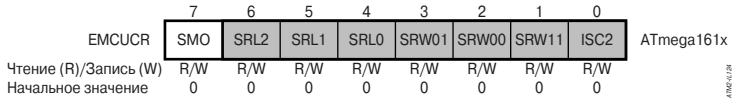


Рис. 2.46. Формат регистра EMCUCR модели ATmega161x

В общей сложности для управления «спящим» режимом в микроконтроллерах семейства используется три или четыре (в зависимости от модели) разряда регистров ввода/вывода. Назначение этих разрядов приведено в **Табл. 2.44**.

Таблица 2.44. **Разряды регистров для управления «спящим» режимом**

Название разряда	Описание
SE	**Разрешение перехода в режим пониженного энергопотребления.** Установка этого разряда в «1» разрешает перевод микроконтроллера в режим пониженного энергопотребления. Переключение осуществляется по команде SLEEP. При сброшенном разряде SE выполнение команды не производит никаких действий
SM2...SM0 (SM1,SM0)	**Выбор режима пониженного энергопотребления.** Состояние этих разрядов определяет, в какой режим перейдет микроконтроллер после выполнения команды SLEEP

Переключение в любой из режимов пониженного потребления осуществляется командой SLEEP. При этом флаг SE должен быть установлен в «1». Во избежание непреднамеренного переключения микроконтроллера в «спящий» режим рекомендуется устанавливать этот флаг непосредственно перед выполнением команды SLEEP. Режим, в который перейдет микроконтроллер после выполнения команды SLEEP, определяется состоянием разрядов SM2...SM0 (SM1, SM0 для моделей ATmega162x и ATmega163x). Соответствие между содержимым этих разрядов и режимом пониженного энергопотребления приведено в **Табл. 2.45**.

Таблица 2.45. **Выбор режима пониженного энергопотребления**

SM2*	SM1	SM0	Режим
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power Down
0	1	1	Power Save
1	0	0	Зарезервировано
1	0	1	Зарезервировано
1	1	0	Standby**
1	1	1	Extended Standby**

* В регистрах моделей ATmega161x и ATmega163x этот разряд отсутствует.
** Эти режимы доступны только при использовании в качестве источника тактового сигнала кварцевого генератора (1-й режим работы тактового генератора согласно **Табл. 2.32**).

Наличие того или иного «спящего» режима в конкретной модели можно определить по **Табл. 2.42**. При отсутствии в конкретной модели

микроконтроллера какого-либо из режимов соответствующие им значения разрядов SM2...SM0 являются зарезервированными. Выход из «спящего» режима может быть осуществлен в результате прерывания и сброса.

При генерации прерывания микроконтроллер переходит в рабочий режим, останавливается на 4 машинных цикла, выполняет подпрограмму обработки прерывания и возобновляет выполнение программы с инструкции, следующей за командой SLEEP. Содержимое РОН, ОЗУ и РВВ при этом не изменяется.

После перехода микроконтроллера в рабочий режим управление передается по адресу вектора сброса.

Idle (ждущий режим)

В режиме Idle прекращается формирование тактовых сигналов clk_{CPU} и clk_{FLASH}. При этом ЦПУ микроконтроллера останавливается, а все остальные периферийные устройства (интерфейсные модули, таймеры/счетчики, аналоговый компаратор, АЦП, сторожевой таймер), а также подсистема прерываний продолжают функционировать. Поэтому выход из режима Idle возможен как по внешнему, так и по внутреннему прерыванию. Если разрешена работа АЦП, то преобразование начнет выполняться сразу же после перехода в этот «спящий» режим.

Основным преимуществом режима Idle является быстрая реакция на события, приводящие к «пробуждению» микроконтроллера. Другими словами, выполнение программы начинается сразу же после перехода из режима Idle в рабочий режим.

ADC Noise Reduction (режим снижения шумов АЦП)

Данный режим имеется только в моделях, содержащих в своем составе модуль АЦП. В этом режиме прекращает работу ЦПУ микроконтроллера и подсистема ввода/вывода (отключаются тактовые сигналы clk_{CPU}, clk_{FLASH} и $clk_{I/O}$), а АЦП, подсистема обработки внешних прерываний, сторожевой таймер и блок сравнения адреса модуля TWI продолжают функционировать. За счет этого уменьшаются помехи на входах АЦП, вызываемые работой системы ввода/вывода микроконтроллера, что, в свою очередь, позволяет повысить точность преобразования. Если АЦП включен, преобразование начинается сразу же после перехода в этот «спящий» режим.

Поскольку тактовый сигнал подсистемы ввода/вывода $clk_{I/O}$ в этом режиме не формируется, возврат микроконтроллера в рабочий режим может произойти только в результате сброса (аппаратного, от сторожевого таймера, от схемы BOD) или в результате генерации следующих прерываний:

- прерывания по совпадению адреса от интерфейса TWI;
- внешнего прерывания (обнаруживаемого асинхронно);
- прерывания от EEPROM-памяти и SPM-прерывания;
- прерывания от АЦП.

Power Down (режим микропотребления)

В режиме Power Down отключаются все внутренние тактовые сигналы, соответственно прекращается функционирование всех систем микроконтроллера, работающих в синхронном режиме. Единственными узлами, продолжающими работать в этом режиме, являются асинхронные модули микроконтроллера: сторожевой таймер (если он включен), подсистема обработки внешних прерываний и блок сравнения адреса модуля TWI. Соответственно выход из режима Power Down возможен либо в результате сброса (аппаратного, от сторожевого таймера, от схемы BOD) или в результате генерации прерываний:

- прерывания по совпадению адреса от интерфейса TWI;
- внешнего прерывания (обнаруживаемого асинхронно).

Поскольку тактовый генератор микроконтроллера в режиме Power Down останавливается, между наступлением события, приводящего к «пробуждению» микроконтроллера и началом его работы проходит некоторое время, в течение которого тактовый генератор микроконтроллера выходит на рабочий режим. Эта задержка определяется теми же конфигурационными ячейками, которые определяют задержку сброса микроконтроллера (см. раздел 10.4).

Также следует помнить, что для выхода микроконтроллера из режима Power Down в результате внешнего прерывания, генерируемого по НИЗКОМУ уровню на входе, длительность активного сигнала должна быть не меньше двух периодов сигнала тактового генератора сторожевого таймера (≥2 мкс при $V_{CC} = 5$ В). Причем, если сигнал, вызвавший «пробуждение» микроконтроллера, исчезнет раньше, чем микроконтроллер перейдет в рабочий режим, обработчик соответствующего прерывания вызван не будет!

Power Save (экономичный режим)

Этот режим идентичен режиму Power Down, за одним исключением: если таймер/счетчик микроконтроллера, поддерживающий работу в асинхронном режиме, сконфигурирован для работы в этом режиме, то он будет работать во время «сна» микроконтроллера. Поэтому выход из режима Power Save возможен не только в результате событий, перечисленных при рассмотрении режима Power Down, но и по прерываниям от таймера/счетчика. Разумеется, эти прерывания должны быть разрешены.

Standby (режим ожидания)

Этот режим доступен только при использовании генератора с внешним резонатором в качестве источника тактового сигнала. Режим Standby полностью идентичен режиму Power Down, за исключением того что тактовый генератор продолжает функционировать. Благодаря этому переход микроконтроллера в рабочий режим происходит гораздо быстрее — за 6 машинных циклов.

Extended Standby (расширенный режим ожидания)

Как и режим Standby, этот режим доступен только при использовании генератора с внешним резонатором. Режим Standby полностью идентичен режиму Power Save, за исключением того что тактовый генератор продолжает функционировать. Поэтому интервал между «пробуждением» микроконтроллера и выходом его в рабочий режим составляет всего 6 машинных циклов.

Основные отличия разных режимов пониженного энергопотребления сведены в **Табл. 2.46**.

Таблица 2.46. **Основные отличия режимов пониженного энергопотребления**

Режим	Активные внутренние тактовые сигналы					Генераторы		Источники «пробуждения» микроконтроллера					
	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Основной источник тактового сигнала	Генератор таймера/счетчика	Внешние прерывания	Блок совпадения адреса модуля TWI	Асинхронный таймер/счетчик	Готовность SPM/EEPROM	АЦП	Другие модули ввода/вывода
Idle			♦	♦	♦	♦	♦*	♦	♦	♦	♦	♦	♦
ADC Noise Reduction				♦	♦	♦	♦*	♦**	♦	♦	♦	♦	
Power Down								♦**	♦				
Power Save					♦*		♦*	♦**	♦	♦*			
Standby ***						♦		♦**	♦				
Extended Standby (1)					♦*	♦	♦*	♦**	♦	♦*			

* Если таймер/счетчик работает в асинхронном режиме.
** Только прерывания, обнаруживаемые асинхронно.
*** Источником тактового сигнала должен быть кварцевый генератор.

10.4. Сброс

Реинициализация, или так называемый «сброс», переводит микроконтроллер в определенное устойчивое состояние. Сброс может быть вызван следующими событиями:

- включение напряжения питания микроконтроллера;
- подача сигнала НИЗКОГО уровня на вывод $\overline{\text{RESET}}$ (аппаратный сброс);
- тайм-аут сторожевого таймера;
- падение напряжения питания ниже заданной величины;
- сброс по интерфейсу JTAG.

При наступлении любого из перечисленных событий во все регистры ввода/вывода заносятся их начальные значения, а в счетчик команд загружается значение адреса вектора сброса. По этому адресу должна находиться команда безусловного перехода (RJMP для моделей ATmega8x и ATmega8515x, JMP — для остальных) на начало программы. Если же прерывания в программе не используются, то она может начинаться непосредственно с адреса вектора сброса. Сказанное справедливо и для случая, когда вектор сброса располагается в области основной программы, а таблица векторов прерываний — в области загрузчика.

Значение адреса вектора сброса определяется конфигурационной ячейкой BOOTRST. Если BOOTRST = «1» (до программирования), вектор сброса располагается в самом начале памяти программ по адресу $0000. После программирования ячейки вектор сброса располагается в начале области загрузчика. Конкретное значение этого адреса зависит от установок конфигурационных ячеек BOOTSZ1 и BOOTSZ0 (кроме моделей ATmega161x), определяющих, в том числе, и размер области загрузчика. Подробно использование этих ячеек будет рассмотрено в Главе 26. Зависимость значения адреса вектора сброса от установок конфигурационных ячеек BOOTSZ1 и BOOTSZ0 для всех моделей семейства приведено в **Табл. 2.47**.

Таблица 2.47. **Значения адреса вектора сброса**

BOOTSZ1	BOOTSZ0	ATmega8x ATmega8515x	ATmega16x	ATmega161x *	ATmega162x ATmega163x	ATmega32x	ATmega323x	ATmega64x	ATmega128x
1	1	$F80	$1F80	$1E00	$1F80	$3F00	$3F00	$7E00	$FE00
1	0	$F00	$1F00		$1F00	$3E00	$3E00	$7C00	$FC00
0	1	$E00	$1E00		$1E00	$3C00	$3C00	$7800	$F800
0	0	$C00	$1C00		$1C00	$3800	$3800	$7000	$F000

* В моделях ATmega161x конфигурационные ячейки BOOTSZ1 и BOOTSZ0 отсутствуют.

Обобщенная структурная схема подсистемы сброса приведена на **Рис. 2.47**. Обратите внимание на то, что элементы, нарисованные пунктиром, в ряде моделей отсутствуют.

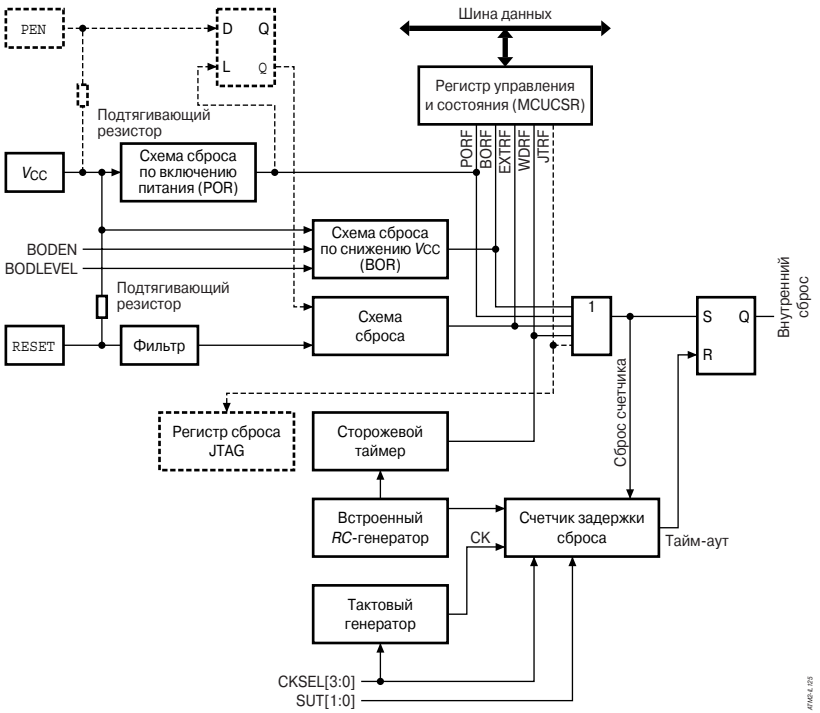


Рис. 2.47. Структурная схема подсистемы сброса микроконтроллеров семейства Mega

Логика схемы сброса всех микроконтроллеров семейства Mega следующая. При наступлении события, приводящего к сбросу микроконтроллера, формируется внутренний сигнал сброса. Одновременно запускается таймер формирования задержки сброса. По истечении определенного промежутка времени внутренний сигнал сброса снимается и начинается выполнение программы.

Все микроконтроллеры семейства позволяют определить событие, в результате которого произошел сброс устройства. Для этой цели используется регистр управления и состояния микроконтроллера MCUCSR, располо-

женный по адресу $34 ($54). Помимо всего прочего, этот регистр содержит набор флагов, состояние которых зависит от события, вызвавшего сброс устройства. Формат регистра MCUCSR всех моделей семейства приведен на **Рис. 2.48** (разряды, не относящиеся к подсистеме сброса, выделены серым цветом). Описание флагов, используемых для определения источника сброса, приведено в **Табл. 2.48**.

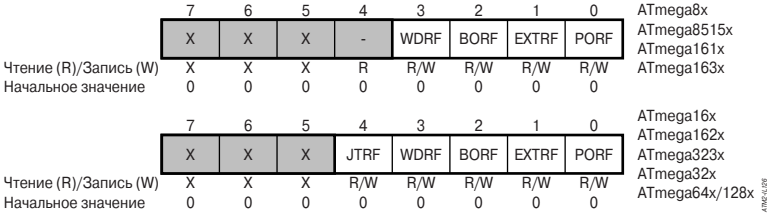


Рис. 2.48. Формат регистра MCUCSR

Таблица 2.48. **Флаги источников сброса регистра MCUCSR**

Название флага	Описание
JTRF	**Флаг JTAG-сброса.** Устанавливается в «1», если сброс произошел в результате команды JTAG «AVR_RESET». Разряд сбрасывается в результате сброса по питанию или непосредственной записью в него лог. 0
WDRF	**Флаг сброса от сторожевого таймера.** Устанавливается в «1», если источником сброса был сторожевой таймер. Разряд сбрасывается в результате сброса по питанию или непосредственной записью в него лог. 0
BORF	**Флаг сброса по снижению питания.** Устанавливается в «1», если источником сброса была подсистема BOD. Разряд сбрасывается в результате сброса по питанию или непосредственной записью в него лог. 0
EXTRF	**Флаг аппаратного сброса.** Устанавливается в «1», если сброс произошел в результате подачи на вывод сброса сигнала НИЗКОГО уровня. Разряд сбрасывается в результате сброса по питанию или непосредственной записью в него лог. 0
PORF	**Флаг сброса по включению питания.** Устанавливается в «1» после подачи напряжения питания на микроконтроллер. Разряд сбрасывается только непосредственной записью в него лог. 0

10.4.1. Сброс по включению питания

В состав всех микроконтроллеров семейства Mega входит система сброса по включению питания (схема POR, Power-on Reset). Эта схема удерживает микроконтроллер в состоянии сброса до тех пор, пока напряжение питания не превысит некоторого порогового значения V_{POT}. При достижении напря-

жением питания значения V_{POT} схема POR запускает таймер задержки сброса. По окончании счета (после формирования задержки t_{TOUT}) внутренний сигнал сброса снимается и происходит запуск микроконтроллера.

Управлением выводом $\overline{\text{RESET}}$ микроконтроллера при включении питания может осуществляться двумя способами. Если время нарастания напряжения источника питания известно и не превышает величины t_{TOUT}, можно использовать первый способ, при котором напряжение на выводе $\overline{\text{RESET}}$ «повторяет» напряжение питания. Соответствующие данному способу временные диаграммы показаны на **Рис. 2.49**. Вывод $\overline{\text{RESET}}$ можно подключить к источнику питания либо оставить неподключенным, поскольку он уже подсоединен к источнику питания подтягивающим резистором.

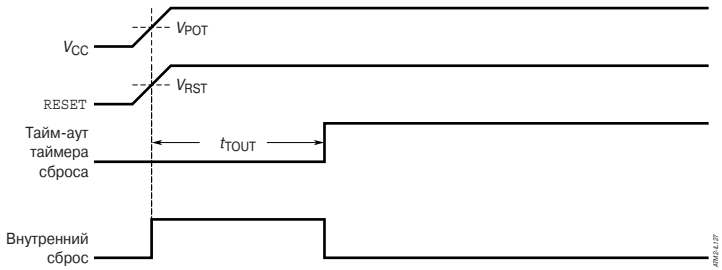


Рис. 2.49. Временные диаграммы сигналов при сбросе в момент включения питания; вывод $\overline{\text{RESET}}$ подключен к V_{DD}

При втором способе управление выводом $\overline{\text{RESET}}$ осуществляется внешней схемой, и сигнал ВЫСОКОГО уровня подается на него только после установления напряжения питания. Временные диаграммы, соответствующие этому способу, показаны на **Рис. 2.50**.

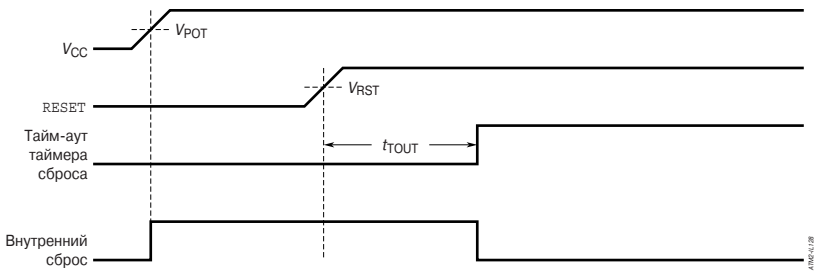


Рис. 2.50. Временные диаграммы сигналов при сбросе по включению питания; вывод $\overline{\text{RESET}}$ управляется внешней схемой

В этом случае работой таймера задержки сброса будет управлять схема аппаратного сброса, и он начнет работать при достижении напряжения на выводе $\overline{\text{RESET}}$ порогового значения V_{RST}.

Данное решение является более дорогостоящим, т. к. требует применения внешних компонентов. Однако этот способ позволяет «подгонять» время запуска микроконтроллера под время нарастания напряжения используемого источника питания.

10.4.2. Аппаратный сброс

Аппаратный (или внешний) сброс микроконтроллера осуществляется подачей на вывод $\overline{\text{RESET}}$ сигнала НИЗКОГО уровня. Микроконтроллер остается в состоянии сброса до тех пор, пока на выводе $\overline{\text{RESET}}$ будет присутствовать сигнал НИЗКОГО уровня. Длительность импульса сброса должна быть не менее 500 нс, в противном случае сброс микроконтроллера не гарантируется. При достижении напряжением на выводе $\overline{\text{RESET}}$ порогового значения V_{RST} запускается таймер задержки сброса. После формирования задержки t_{TOUT} внутренний сигнал сброса снимается и происходит запуск микроконтроллера. Временные диаграммы сигналов при аппаратном сбросе показаны на **Рис. 2.51**.

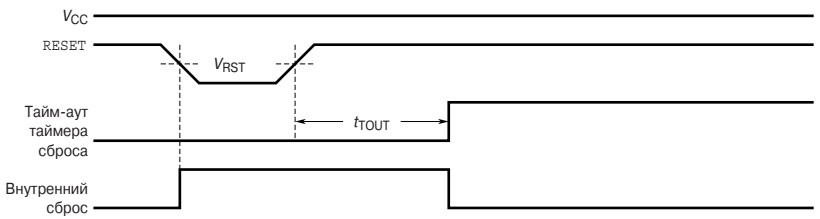


Рис. 2.51. Временные диаграммы сигналов при аппаратном сбросе

10.4.3. Сброс от сторожевого таймера

По тайм-ауту сторожевого таймера (если он включен) генерируется короткий положительный импульс сброса, длительность которого равна одному периоду тактового сигнала микроконтроллера. По спадающему фронту этого импульса запускается таймер задержки сброса. После формирования задержки t_{TOUT} внутренний сигнал сброса снимается и происходит запуск микроконтроллера. Временные диаграммы, соответствующие сбросу от сторожевого таймера, показаны на **Рис. 2.52**.

Рис.2.52. Временные диаграммы сигналов при сбросе от сторожевого таймера

10.4.4. Сброс при снижении напряжения питания

Все модели микроконтроллеров семейства Mega имеют в своем составе схему BOD (Brown-Out Detection), которая отслеживает величину напряжения источника питания. Если работа этой схема разрешена, то при снижении напряжения питания ниже некоторого значения она переводит микроконтроллер в состояние сброса. Когда напряжение питания вновь увеличится до порогового значения, запускается таймер задержки сброса. После формирования задержки t_{TOUT} внутренний сигнал сброса снимается и происходит запуск микроконтроллера. Временные диаграммы, соответствующие сбросу от схемы BOD, показаны на **Рис.2.53**.

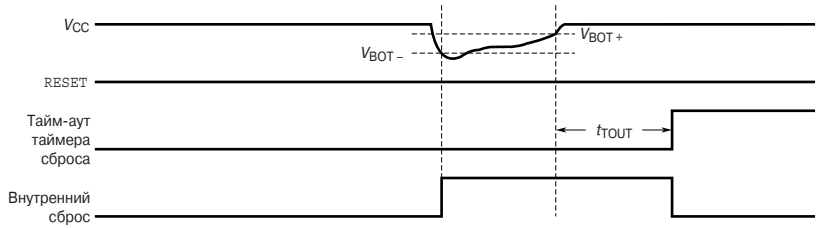


Рис. 2.53. Временные диаграммы сигналов при сбросе по снижению напряжения питания

Во всех моделях, кроме ATmega162x, для управления схемой BOD используются две конфигурационные ячейки. Включением/выключением схемы BOD управляет конфигурационная ячейка BODEN. Для разрешения работы схемы эта ячейка должна быть запрограммирована («0»). Порог срабатывания (V_{BOT}) определяется состоянием конфигурационной ячейки BODLEVEL. Если в этой ячейке записана лог. 1, порог срабатывания

равен 2.7 В. Если же в ней записан лог. 0 (после ее программирования), порог срабатывания равен 4.0 В.

В моделях ATmega162x для управления схемой BOD используются уже три конфигурационных ячейки BODLEVEL2...0. Влияние содержимого этих ячеек на работу схемы BOD приведено в **Табл. 2.49**.

Таблица 2.49. **Конфигурационные ячейки BODLEVEL2...0 (ATmega162x)**

BODLEVEL2...0	Порог срабатывания V_{BOT} [В]
111	Схема BOD выключена
110	1.8
101	2.7
100	4.0
011	Зарезервировано
010	
001	
000	

Для уменьшения вероятности ложных срабатываний порог напряжения переключения схемы имеет гистерезис, равный 50 мВ ($V_{BOT+} = V_{BOT} + 25$ мВ, $V_{BOT-} = V_{BOT} - 25$ мВ). Кроме того, срабатывание схемы BOD произойдет только в том случае, если длительность провала напряжения питания будет больше некоторой величины. Для моделей ATmega161x, ATmega163x и ATmega323x эта величина составляет 9 мкс при $V_{BOT} = 4.0$ В и 21 мкс при $V_{BOT} = 2.7$ В. Для остальных моделей минимальная длительность провала составляет 2 мкс для любого режима работы схемы BOD.

10.4.5. Управление схемой сброса

Управление схемой сброса заключается в задании длительности задержки сброса t_{TOUT}. Для этого используются те же конфигурационные ячейки, которые определяют режим работы тактового генератора микроконтроллера используются еще и в некоторых моделях другие конфигурационные ячейки.

Собственно задержка сброса имеет две составляющие. В течение первой части задержки (t_S) происходит выход на рабочий режим и стабилизация частоты тактового генератора перед началом выполнения команд. Кроме того, эта составляющая определяет длительность перехода микроконтроллера в рабочий режим из режимов Power Down и Power Save. При формировании этой составляющей таймер задержки сброса работает от тактового генератора микроконтроллера.

Вторая часть задержки (t_R) предназначена для того, чтобы дать возможность установиться напряжению питания. При формировании этой составляющей задержки таймер работает от *RC*-генератора сторожевого таймера.

ATmega8x/8515x/16x/32x/64x/128x

Для управления длительностью задержки сброса используется конфигурационная ячейка CKSEL0 (младший разряд числа, определяющего режим работы тактового генератора), а также две дополнительные конфигурационные ячейки SUT1 и SUT0. Задание длительности задержки сброса для каждого из режимов работы тактового генератора показано в **Табл. 2.50**.

Таблица 2.50. **Определение задержки сброса ATmega8x/8515x/16x/64x/128x**

Режим работы тактового генератора	CKSEL0	SUT1...0	t_S [такты]	t_R (V_{CC} = 5.0 В) [мс]	Рекомендуемое использование	Примечание
Кварцевый или керамический резонатор (1111...1010)	0	00	258	4	Керамический резонатор, малое время нарастания напряжения питания	1
	0	01	258	64	Керамический резонатор, большое время нарастания напряжения питания	
	0	10	1K	—	Керамический резонатор, схема BOD включена	2
	0	11	1K	4	Керамический резонатор, малое время нарастания напряжения питания	
	1	00	1K	64	Керамический резонатор, большое время нарастания напряжения питания	
	1	01	16K	—	Кварцевый резонатор, схема BOD включена	—
	1	10	16K	4	Кварцевый резонатор, малое время нарастания напряжения питания	
	1	11	16K	64	Кварцевый резонатор, большое время нарастания напряжения питания	
Низкочастотный кварцевый резонатор (1001)	—	00	1K	4	Малое время нарастания напряжения питания или схема BOD включена	3

Продолжение таблицы 2.50

Режим работы тактового генератора	CKSEL0	SUT1...0	t_S [такты]	t_R (V_{CC} = 5.0 В) [мс]	Рекомендуемое использование	Примечание
Низкочастотный кварцевый резонатор (1001)	—	01	1K	64	Большое время нарастания напряжения питания	3
	—	10	32K	64	Необходимость стабильной частоты при старте программы	
	—	11			Зарезервировано	
Встроенный генератор с внешней *RC*-цепочкой (1000...0101)	—	00	18	—	Схема BOD включена	—
	—	01	18	4	Малое время нарастания напряжения питания	
	—	10	18	64	Большое время нарастания напряжения питания	
	—	11	6	4	Малое время нарастания напряжения питания или схема BOD включена	4
Встроенный генератор с внутренней *RC*-цепочкой (0100...0001)	—	00	6	—	Схема BOD включена	—
	—	01	6	4	Малое время нарастания напряжения питания	
	—	10	6	64	Большое время нарастания напряжения питания	5
	—	11			Зарезервировано	
Внешний сигнал синхронизации (0000)	—	00	6	—	Схема BOD включена	—
	—	01	6	4	Малое время нарастания напряжения питания	
	—	10	6	64	Большое время нарастания напряжения питания	
	—	11			Зарезервировано	

Примечания: 1. Могут быть использованы только при работе на частотах, далеких от максимальной, и при отсутствии жестких требований к стабильности тактовой частоты при старте программы.
2. Могут быть использованы также для кварцевых резонаторов при работе на частотах, далеких от максимальной, и при отсутствии жестких требований к стабильности тактовой частоты при старте программы.
3. Могут быть использованы только в том случае, если стабильность тактовой частоты при старте программы не требуется.
4. Не должны использоваться при работе на частотах, близких к максимальной.
5. Состояние по умолчанию.

ATmega161x

В этих моделях длительность задержки сброса определяется состоянием конфигурационных ячеек CKSEL2...0 и BODLEVEL. Причем действие ячейки BODLEVEL не зависит от того, включена схема BOD или нет. Задание длительности задержки сброса для этих моделей показано в **Табл. 2.51**. При подаче напряжения питания составляющая t_R задержки сброса увеличивается примерно на 0.6 мс.

Таблица 2.51. **Определение задержки сброса в моделях ATmega161x**

CKSEL 2...0	t_S [такты]	t_R [мс]		Рекомендуемое использование
		V_{CC} = 2.7 В, BODLEVEL = «1»	V_{CC} = 4.0 В, BODLEVEL = «0»	
000	6	4.2	5.8	Внешний сигнал синхронизации, малое время нарастания напряжения питания
001	6	30*	10*	Внешний сигнал синхронизации, схема BOD включена
010	16K	67	92	Кварцевый резонатор, большое время нарастания напряжения питания
011	16K	4.2	5.8	Кварцевый резонатор, малое время нарастания напряжения питания
100	16K	30*	10*	Кварцевый резонатор, схема BOD включена
101	1K	67	92	Керамический резонатор или внешний сигнал синхронизации, большое время нарастания напряжения питания
110	1K	4.2	5.8	Керамический резонатор, малое время нарастания напряжения питания
111	1K	30*	10*	Керамический резонатор, схема BOD включена

* мкс; при включенной схеме BOD $t_R \approx 50$ мкс.

ATmega162x

Для управления длительностью задержки сброса используются три конфигурационные ячейки: CKSEL0 и SUT1...0. Задание длительности задержки сброса для каждого из режимов работы тактового генератора показано в **Табл. 2.52**.

Таблица 2.52. **Определение задержки сброса в моделях ATmega162x**

Режим работы тактового генератора	CKSEL0	SUT1...0	t_S [такты]	t_R (V_{CC} = 5.0 В) [мс]	Рекомендуемое использование	Примечание
Кварцевый или керамический резонатор (1111...1000)	0	00	258	4	Керамический резонатор, малое время нарастания напряжения питания	1
	0	01	258	64	Керамический резонатор, большое время нарастания напряжения питания	
	0	10	1K	—	Керамический резонатор, схема BOD включена	2
	0	11	1K	4	Керамический резонатор, малое время нарастания напряжения питания	
	1	00	1K	64	Керамический резонатор, большое время нарастания напряжения питания	
	1	01	16K	—	Кварцевый резонатор, схема BOD включена	—
	1	10	16K	4	Кварцевый резонатор, малое время нарастания напряжения питания	
	1	11	16K	64	Кварцевый резонатор, большое время нарастания напряжения питания	
Низкочастотный кварцевый резонатор (0111...0100)	0	00	1K	0	Малое время нарастания напряжения питания или схема BOD включена	3
	0	01	1K	4		
	0	10	1K	64	Большое время нарастания напряжения питания	
	0	11	—	—	Зарезервировано	—
	1	00	32K	0	Малое время нарастания напряжения питания или схема BOD включена	

Продолжение таблицы 2.52

Режим работы тактового генератора	CKSEL0	SUT1...0	t_S [такты]	t_R (V_{CC} = 5.0 В) [мс]	Рекомендуемое использование	Примечание
Низкочастотный кварцевый резонатор (0111...0100)	1	01	32K	4		—
	1	10	32K	64	Большое время нарастания напряжения питания	
	1	11	—	—	Зарезервировано	
Внутренний *RC*-генератор (0010) Внешний сигнал синхронизации (0000)	—	00	6	—	Схема BOD включена	—
	—	01	6	4	Малое время нарастания напряжения питания	
	—	10	6	64	Большое время нарастания напряжения питания	4
	—	11	—	—	Зарезервировано	—

Примечания: 1. Могут быть использованы только при работе на частотах, далеких от максимальной, и при отсутствии жестких требований к стабильности тактовой частоты при старте программы.
2. Могут быть использованы также для кварцевых резонаторов при работе на частотах, далеких от максимальной, и при отсутствии жестких требований к стабильности тактовой частоты при старте программы.
3. Могут быть использованы только в том случае, если стабильность тактовой частоты при старте программы не требуется.
4. Состояние по умолчанию.

ATmega163x, ATmega323x

В этих моделях длительность задержки сброса определяется состоянием конфигурационных ячеек CKSEL3...0 и BODLEVEL. Действие ячейки BODLEVEL не зависит от того, включена схема BOD или нет. Задание длительности задержки сброса для этих моделей показано в **Табл. 2.53**. При подаче напряжения питания составляющая t_R задержки сброса увеличивается примерно на 0.6 мс.

Таблица 2.53. **Определение задержки сброса в моделях ATmega163/323x**

CKSEL 3:0	t_S [такты]	t_R [мс]		Рекомендуемое использование
		V_{CC} = 2.7 В, BODLEVEL = «1»	V_{CC} = 4.0 В, BODLEVEL = «0»	
0000	6	4.2	5.8	Внешний сигнал синхронизации, малое время нарастания напряжения питания
0001	6	30*	10*	Внешний сигнал синхронизации, схема BOD включена
0010**	6	67	92	Внешняя *RC*-цепочка, большое время нарастания напряжения питания
0011	6	4.2	5.8	Внешняя *RC*-цепочка, малое время нарастания напряжения питания
0100	6	30*	10*	Внешняя *RC*-цепочка, схема BOD включена
0101	6	67	92	Внутренняя *RC*-цепочка, большое время нарастания напряжения питания
0110	6	4.2	5.8	Внутренняя *RC*-цепочка, малое время нарастания напряжения питания
0111	6	30*	10*	Внутренняя *RC*-цепочка, схема BOD включена
1000	32K	67	92	Низкочастотный кварцевый резонатор
1001	1K	67	92	Низкочастотный кварцевый резонатор
1010	16K	67	92	Кварцевый резонатор, большое время нарастания напряжения питания
1011	16K	4.2	5.8	Кварцевый резонатор, малое время нарастания напряжения питания
1100	16K	30*	10*	Кварцевый резонатор, схема BOD включена
1101	1K	67	92	Керамический резонатор или внешний сигнал синхронизации, большое время нарастания напряжения питания
1110	1K	4.2	5.8	Керамический резонатор, малое время нарастания напряжения питания
1111	1K	30*	10*	Керамический резонатор, схема BOD включена

* мкс; при включенной схеме BOD $t_R \approx 130$ мкс при BODLEVEL = «1» и $t_R \approx 35$ мкс при BODLEVEL = «0».
** Значение по умолчанию.

Действительное значение задержек может отличаться от приведенных в **Табл. 2.48...2.51**, поскольку для формирования составляющей t_R задержки используется *RC*-генератор сторожевого таймера, на частоту которого влияют различные факторы, в частности величина напряжения питания.

Глава 11. **Прерывания**

11.1. 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 обеспечивает возврат в основную программу и восстановление предварительно сохраненного счетчика команд.

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

11.2. Таблица векторов прерываний

Как и все микроконтроллеры AVR, микроконтроллеры семейства Mega имеют многоуровневую систему приоритетных прерываний. Младшие адреса памяти программ начиная с адреса $0001 (модели ATmega8x и ATmega8515x) и $0002 (остальные модели) отведены под таблицу векторов прерывания. Каждому прерыванию соответствует адрес в этой таблице, который загружается в счетчик команд при возникновении прерывания. Положение вектора в таблице также определяет и приоритет соответствующего прерывания: чем меньше адрес, тем выше приоритет прерывания. Размер вектора прерывания зависит от объема памяти программ микроконтроллера и составляет 1 байт для моделей ATmega8x и ATmega8515x и 2 байта для остальных моделей. Соответственно для перехода к подпрограммам обработки прерываний в моделях микроконтроллеров

ATmega8x и ATmega8515x используются команды RJMP, а в остальных моделях — команды JMP.

Практически во всех микроконтроллерах семейства Mega, за исключением моделей ATmega161x и ATmega163x, положение таблицы векторов прерываний может быть изменено. Таблица может располагаться не только в начале памяти программ, а также и в начале области загрузчика. Причем перемещение таблицы может быть осуществлено непосредственно в ходе выполнения программы.

Управление размещением таблицы прерываний осуществляется двумя младшими разрядами регистров MCUCR (модели ATmega64x и ATmega128x) и GICR (остальные модели): IVSEL (1-й разряд) и IVCE (0-й разряд). Состояние флага IVSEL определяет положение таблицы в памяти программ. Если флаг сброшен в «0», таблица векторов прерываний располагается в начале памяти программ, если флаг установлен в «1» — в начале области загрузчика. Конкретное значение начального адреса области загрузчика зависит от установок конфигурационных ячеек BOOTSZ1 и BOOTSZ0 (кроме моделей ATmega161x). Разряд IVCE предназначен для разрешения изменения флага IVSEL.

Для изменения положения таблицы векторов прерываний необходимо установить разряд IVCE в «1» и затем в течение следующих четырех машинных циклов занести требуемое значение в разряд IVSEL. При этом разряд IVCE сбрасывается в «0». В противном случае разряд IVCE будет сброшен аппаратно по истечении четырех машинных циклов, 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 34 (модели ATmega64x и ATmega128x) векторов. Распределение адресов таблицы векторов прерываний для различных микроконтроллеров семейства приведено в **Табл. 2.54...2.62**. При размещении векторов прерываний в области загрузчика к значениям, указанным в таблицах, следует прибавить значение начального адреса области загрузчика. Обратите внимание на то, что таблица векторов прерываний моделей ATmega162x зависит от состояния конфигурационной ячейки M161C, определяющей режим функционирования микроконтроллера. При запрограммированной ячейке (в режиме совместимости с ATmega161x) количество векторов прерываний уменьшается, поскольку часть периферийных устройств в этом режиме не поддерживается.

Таблица 2.54. **Таблица векторов прерываний моделей ATmega8x**

Источник	Описание	№	Адрес
INT0	Внешнее прерывание 0	1	$0001
INT1	Внешнее прерывание 1	2	$0002
TIMER2 COMP	Совпадение таймера/счетчика T2	3	$0003
TIMER2 OVF	Переполнение таймера/счетчика T2	4	$0004
TIMER1 CAPT	Захват таймера/счетчика T1	5	$0005
TIMER1 COMPA	Совпадение «A» таймера/счетчика T1	6	$0006
TIMER1 COMPB	Совпадение «B» таймера/счетчика T1	7	$0007
TIMER1 OVF	Переполнение таймера/счетчика T1	8	$0008
TIMER0 OVF	Переполнение таймера/счетчика T0	9	$0009
SPI, STC	Передача по SPI завершена	10	$000A
USART, RXC	USART, прием завершен	11	$000B
USART, UDRE	Регистр данных USART пуст	12	$000C
USART, TXC	USART, передача завершена	13	$000D
ADC	Преобразование АЦП завершено	14	$000E
EE_RDY	EEPROM, готово	15	$000F
ANA_COMP	Аналоговый компаратор	16	$0010
TWI	Прерывание от модуля TWI	17	$0011
SPM_RDY	Готовность SPM	18	$0012

Таблица 2.55. **Таблица векторов прерываний моделей ATmega8515x**

Источник	Описание	№	Адрес
INT0	Внешнее прерывание 0	1	$0001
INT1	Внешнее прерывание 1	2	$0002
TIMER1 CAPT	Захват таймера/счетчика T1	3	$0003
TIMER1 COMPA	Совпадение «A» таймера/счетчика T1	4	$0004
TIMER1 COMPB	Совпадение «B» таймера/счетчика T1	5	$0005
TIMER1 OVF	Переполнение таймера/счетчика T1	6	$0006
TIMER0 OVF	Переполнение таймера/счетчика T0	7	$0007
SPI, STC	Передача по SPI завершена	8	$0008
USART, RXC	USART, прием завершен	9	$0009
USART, UDRE	Регистр данных USART пуст	10	$000A
USART, TXC	USART, передача завершена	11	$000B
ANA_COMP	Аналоговый компаратор	12	$000C
INT2	Внешнее прерывание 2	13	$000D
TIMER0 COMP	Совпадение таймера/счетчика T0	14	$000E
EE_RDY	EEPROM, готово	15	$000F
SPM_RDY	Готовность SPM	16	$0010

Таблица 2.56. **Таблица векторов прерываний моделей ATmega16x**

Источник	Описание	№	Адрес
INT0	Внешнее прерывание 0	1	$0002
INT1	Внешнее прерывание 1	2	$0004
TIMER2 COMP	Совпадение таймера/счетчика T2	3	$0006
TIMER2 OVF	Переполнение таймера/счетчика T2	4	$0008
TIMER1 CAPT	Захват таймера/счетчика T1	5	$000A
TIMER1 COMPA	Совпадение «A» таймера/счетчика T1	6	$000C
TIMER1 COMPB	Совпадение «B» таймера/счетчика T1	7	$000E
TIMER1 OVF	Переполнение таймера/счетчика T1	8	$0010
TIMER0 OVF	Переполнение таймера/счетчика T0	9	$0012
SPI, STC	Передача по SPI завершена	10	$0014
USART, RXC	USART, прием завершен	11	$0016
USART, UDRE	Регистр данных USART пуст	12	$0018
USART, TXC	USART, передача завершена	13	$001A
ADC	Преобразование АЦП завершено	14	$001C
EE_RDY	EEPROM, готово	15	$001E
ANA_COMP	Аналоговый компаратор	16	$0020
TWI	Прерывание от модуля TWI	17	$0022
INT2	Внешнее прерывание 2	18	$0024
TIMER0 COMP	Совпадение таймера/счетчика T0	19	$0026
SPM_RDY	Готовность SPM	20	$0028

Таблица 2.57. **Таблица векторов прерываний моделей ATmega161x**

Источник	Описание	№	Адрес
INT0	Внешнее прерывание 0	1	$0002
INT1	Внешнее прерывание 1	2	$0004
INT2	Внешнее прерывание 2	3	$0006
TIMER2 COMP	Совпадение таймера/счетчика T2	4	$0008
TIMER2 OVF	Переполнение таймера/счетчика T2	5	$000A
TIMER1 CAPT	Захват таймера/счетчика T1	6	$000C
TIMER1 COMPA	Совпадение «A» таймера/счетчика T1	7	$000E
TIMER1 COMPB	Совпадение «B» таймера/счетчика T1	8	$0010
TIMER1 OVF	Переполнение таймера/счетчика T1	9	$0012
TIMER0 COMP	Совпадение таймера/счетчика T0	10	$0014
TIMER0 OVF	Переполнение таймера/счетчика T0	11	$0016
SPI, STC	Передача по SPI завершена	12	$0018
UART0, RXC	UART0, прием завершен	13	$001A
UART1, RXC	UART1, прием завершен	14	$001C
UART0, UDRE	Регистр данных UART0 пуст	15	$001E
UART1, UDRE	Регистр данных UART1 пуст	16	$0020
UART0, TXC	UART0, передача завершена	17	$0022
UART1, TXC	UART1, передача завершена	18	$0024
EE_RDY	EEPROM, готово	19	$0026
ANA_COMP	Аналоговый компаратор	20	$0028

Таблица 2.58. **Таблица векторов прерываний моделей ATmega162x**

Источник	Описание	M161C = «1»		M161C = «0»	
		№	Адрес	№	Адрес
INT0	Внешнее прерывание 0	1	$0002	1	$0002
INT1	Внешнее прерывание 1	2	$0004	2	$0004
INT2	Внешнее прерывание 2	3	$0006	3	$0006
PCINT0	Прерывание 0 по изменению состояния вывода	4	$0008	—	—
PCINT1	Прерывание 1 по изменению состояния вывода	5	$000A	—	—
TIMER3 CAPT	Захват таймера/счетчика T3	6	$000C	—	—
TIMER3 COMPA	Совпадение «A» таймера/счетчика T3	7	$000E	—	—
TIMER3 COMPB	Совпадение «B» таймера/счетчика T3	8	$0010	—	—
TIMER3 OVF	Переполнение таймера/счетчика T3	9	$0012	—	—
TIMER2 COMP	Совпадение таймера/счетчика T2	10	$0014	4	$0008
TIMER2 OVF	Переполнение таймера/счетчика T2	11	$0016	5	$000A
TIMER1 CAPT	Захват таймера/счетчика T1	12	$0018	6	$000C
TIMER1 COMPA	Совпадение «A» таймера/счетчика T1	13	$001A	7	$000E
TIMER1 COMPB	Совпадение «B» таймера/счетчика T1	14	$001C	8	$0010
TIMER1 OVF	Переполнение таймера/счетчика T1	15	$001E	9	$0012
TIMER0 COMP	Совпадение таймера/счетчика T0	16	$0020	10	$0014
TIMER0 OVF	Переполнение таймера/счетчика T0	17	$0022	11	$0016
SPI, STC	Передача по SPI завершена	18	$0024	12	$0018
USART0, RXC	USART0, прием завершен	19	$0026	13	$001A
USART1, RXC	USART1, прием завершен	20	$0028	14	$001C
USART0, UDRE	Регистр данных USART0 пуст	21	$002A	15	$001E
USART1, UDRE	Регистр данных USART1 пуст	22	$002C	16	$0020
USART0, TXC	USART0, передача завершена	23	$002E	17	$0022
USART1, TXC	USART1, передача завершена	24	$0030	18	$0024
EE_RDY	EEPROM, готово	25	$0032	19	$0026
ANA_COMP	Аналоговый компаратор	26	$0034	20	$0028
SPM_RDY	Готовность SPM	27	$0036	21	$002A

Таблица 2.59. **Таблица векторов прерываний моделей ATmega163x**

Источник	Описание	№	Адрес
INT0	Внешнее прерывание 0	1	$0002
INT1	Внешнее прерывание 1	2	$0004
TIMER2 COMP	Совпадение таймера/счетчика T2	3	$0006
TIMER2 OVF	Переполнение таймера/счетчика T2	4	$0008

Продолжение таблицы 2.59

Источник	Описание	№	Адрес
TIMER1 CAPT	Захват таймера/счетчика T1	5	$000A
TIMER1 COMPA	Совпадение «A» таймера/счетчика T1	6	$000C
TIMER1 COMPB	Совпадение «B» таймера/счетчика T1	7	$000E
TIMER1 OVF	Переполнение таймера/счетчика T1	8	$0010
TIMER0 OVF	Переполнение таймера/счетчика T0	9	$0012
SPI, STC	Передача по SPI завершена	10	$0014
UART, RXC	UART, прием завершен	11	$0016
UART, UDRE	Регистр данных UART пуст	12	$0018
UART, TXC	UART, передача завершена	13	$001A
ADC	Преобразование АЦП завершено	14	$001C
EE_RDY	EEPROM, готово	15	$001E
ANA_COMP	Аналоговый компаратор	16	$0020
TWI	Прерывание от модуля TWI	17	$0022

Таблица 2.60. **Таблица векторов прерываний моделей ATmega32x**

Источник	Описание	№	Адрес
INT0	Внешнее прерывание 0	1	$0002
INT1	Внешнее прерывание 1	2	$0004
INT2	Внешнее прерывание 2	3	$0006
TIMER2 COMP	Совпадение таймера/счетчика T2	4	$0008
TIMER2 OVF	Переполнение таймера/счетчика T2	5	$000A
TIMER1 CAPT	Захват таймера/счетчика T1	6	$000C
TIMER1 COMPA	Совпадение «A» таймера/счетчика T1	7	$000E
TIMER1 COMPB	Совпадение «B» таймера/счетчика T1	8	$0010
TIMER1 OVF	Переполнение таймера/счетчика T1	9	$0012
TIMER0 COMP	Совпадение таймера/счетчика T0	10	$0014
TIMER0 OVF	Переполнение таймера/счетчика T0	11	$0016
SPI, STC	Передача по SPI завершена	12	$0018
USART, RXC	USART, прием завершен	13	$001A
USART, UDRE	Регистр данных USART пуст	14	$001C
USART, TXC	USART, передача завершена	15	$001E
ADC	Преобразование АЦП завершено	16	$0020
EE_RDY	EEPROM, готово	17	$0022
ANA_COMP	Аналоговый компаратор	18	$0024
TWI	Прерывание от модуля TWI	19	$0026
SPM_RDY	Готовность SPM	20	$0028

Таблица 2.61. **Таблица векторов прерываний моделей ATmega323x**

Источник	Описание	№	Адрес
INT0	Внешнее прерывание 0	1	$0002
INT1	Внешнее прерывание 1	2	$0004
INT2	Внешнее прерывание 2	3	$0006
TIMER2 COMP	Совпадение таймера/счетчика T2	4	$0008
TIMER2 OVF	Переполнение таймера/счетчика T2	5	$000A
TIMER1 CAPT	Захват таймера/счетчика T1	6	$000C
TIMER1 COMPA	Совпадение «A» таймера/счетчика T1	7	$000E
TIMER1 COMPB	Совпадение «B» таймера/счетчика T1	8	$0010
TIMER1 OVF	Переполнение таймера/счетчика T1	9	$0012
TIMER0 COMP	Совпадение таймера/счетчика T0	10	$0014
TIMER0 OVF	Переполнение таймера/счетчика T0	11	$0016
SPI, STC	Передача по SPI завершена	12	$0018
USART, RXC	USART, прием завершен	13	$001A
USART, UDRE	Регистр данных USART пуст	14	$001C
USART, TXC	USART, передача завершена	15	$001E
ADC	Преобразование АЦП завершено	16	$0020
EE_RDY	EEPROM, готово	17	$0022
ANA_COMP	Аналоговый компаратор	18	$0024
TWI	Прерывание от модуля TWI	19	$0026

Таблица 2.62. **Таблица векторов прерываний моделей ATmega64x и ATmega128x**

Источник	Описание	№	Адрес
INT0	Внешнее прерывание 0	1	$0002
INT1	Внешнее прерывание 1	2	$0004
INT2	Внешнее прерывание 2	3	$0006
INT3	Внешнее прерывание 3	4	$0008
INT4	Внешнее прерывание 4	5	$000A
INT5	Внешнее прерывание 5	6	$000C
INT6	Внешнее прерывание 6	7	$000E
INT7	Внешнее прерывание 7	8	$0010
TIMER2 COMP	Совпадение таймера/счетчика T2	9	$0012
TIMER2 OVF	Переполнение таймера/счетчика T2	10	$0014
TIMER1 CAPT	Захват таймера/счетчика T1	11	$0016
TIMER1 COMPA	Совпадение «A» таймера/счетчика T1	12	$0018
TIMER1 COMPB	Совпадение «B» таймера/счетчика T1	13	$001A
TIMER1 OVF	Переполнение таймера/счетчика T1	14	$001C
TIMER0 COMP	Совпадение таймера/счетчика T0	15	$001E
TIMER0 OVF	Переполнение таймера/счетчика T0	16	$0020
SPI, STC	Передача по SPI завершена	17	$0022
USART0, RX	USART0, прием завершен	18	$0024

Продолжение таблицы 2.62

Источник	Описание	№	Адрес
USART0, UDRE	Регистр данных USART0 пуст	19	$0026
USART0, TX	USART0, передача завершена	20	$0028
ADC	Преобразование АЦП завершено	21	$002A
EE_RDY	EEPROM, готово	22	$002C
ANA_COMP	Аналоговый компаратор	23	$002E
TIMER1 COMPC	Совпадение «C» таймера/счетчика T1	24	$0030
TIMER3 CAPT	Захват таймера/счетчика T3	25	$0032
TIMER3 COMPA	Совпадение «A» таймера/счетчика T3	26	$0034
TIMER3 COMPB	Совпадение «B» таймера/счетчика T3	27	$0036
TIMER3 COMPC	Совпадение «C» таймера/счетчика T3	28	$0038
TIMER3 OVF	Переполнение таймера/счетчика T3	29	$003A
USART1, RX	USART1, прием завершен	30	$003C
USART1, UDRE	Регистр данных USART1 пуст	31	$003E
USART1, TX	USART1, передача завершена	32	$0040
TWI	Прерывание от модуля TWI	33	$0042
SPM_RDY	Готовность SPM	34	$0044

Если прерывания в работе микроконтроллера не предусматриваются, то на месте таблицы векторов прерываний может быть размещена часть основной программы. Далее приведены фрагменты листингов для различных положений векторов сброса и прерываний (на примере моделей ATmega128x, размер области загрузчика — 8 Кбайт).

1. Вектор сброса и таблица векторов прерываний располагаются в начале памяти программ (BOOTRST = «1», IVSEL = «0»):

```
Address    Labels    Code                 Comments
$0000                jmp RESET            ;Обработчик сброса
$0002                jmp EXT_INT0         ;Обработчик IRQ0
$0004                jmp EXT_INT1         ;Обработчик IRQ1
...                  ...                  ;
$0044                jmp SPM_RDY          ;Обработчик прерывания
...                  ...                  ;"готовность SPM"
$0046      RESET:    ldi r16,high(RAMEND) ;Начало основной
                                          ;программы
$0047                out SPH,r16
$0048                ldi r16,low(RAMEND)
$0049                out SPH,r16          ;Проинициализировали
                                          ;указатель стека
$004A                sei                  ;Разрешили прерывания
$004B                <команда> xxx
...                  ...
```

2. Вектор сброса располагается в начале памяти программ, а таблица векторов прерываний — в начале области загрузчика (BOOTRST = «1», IVSEL = «1»):

```
Address      Labels    Code                 Comments
$0000        RESET:    ldi r16,high(RAMEND) ;Начало основной
                                            ;программы
$0001                  out SPH,r16
$0002                  ldi r16,low(RAMEND)
$0003                  out SPH,r16          ;Проинициализировали
                                            ;указатель стека
$0004                  sei                  ;Разрешили прерывания
$0005                  <команда> xxx
...                    ...
.org $F002
$F002                  jmp EXT_INT0         ;Обработчик IRQ0
$F004                  jmp EXT_INT1         ;Обработчик IRQ1
...                    ...                  ...
$F044                  jmp SPM_RDY          ;Обработчик прерывания
                                            ;"готовность SPM"
```

3. Вектор сброса располагается в начале области загрузчика, а таблица векторов прерываний — в начале памяти программ (BOOTRST = «0», IVSEL = «0»):

```
Address     Labels   Code                 Comments
.org $0002
$0002                jmp EXT_INT0         ;Обработчик IRQ0
$0004                jmp EXT_INT1         ;Обработчик IRQ1
...                  ...                  ;
$0044                jmp SPM_RDY          ;Обработчик прерывания
                                          ;"готовность SPM"
...                  ...
.org $F000
$F000       RESET:   ldi r16,high(RAMEND) ;Начало основной
                                          ;программы
$F001                out SPH,r16
$F002                ldi r16,low(RAMEND)
$F003                out SPH,r16          ;Проинициализировали
                                          ;указатель стека
$F004                sei                  ;Разрешили прерывания
$F005                <команда> xxx
```

4. Вектор сброса и таблица векторов прерываний располагаются в области загрузчика (BOOTRST = «0», IVSEL = «1»):

```
Address     Labels    Code                 Comments
.org $F000
$F000                 jmp RESET            ;Обработчик сброса
$F002                 jmp EXT_INT0         ;Обработчик IRQ0
$F004                 jmp EXT_INT1         ;Обработчик IRQ1
...                   ...
$F044                 jmp SPM_RDY          ;Обработчик прерывания
...                   ...                  ;"готовность SPM"
$F046       RESET:    ldi r16,high(RAMEND) ;Начало основной
                                           ;программы
$F047                 out SPH,r16
$F048                 ldi r16,low(RAMEND)
$F049                 out SPH,r16          ;Проинициализировали
                                           ;указатель стека
$F04A                 sei                  ;Разрешили прерывания
$F04B                 <команда> xxx
```

11.3. Обработка прерываний

Для глобального разрешения/запрещения прерываний предназначен флаг I регистра SREG. Для разрешения прерываний он должен быть установлен в «1», а для запрещения сброшен в «0». Индивидуальное разрешение или запрещение (маскирование) прерываний производится установкой/сбросом соответствующих разрядов регистров масок прерываний, рассматриваемых 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 установить 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. Прерывания первого типа генерируются при наступлении некоторого события, в результате которого устанавливается флаг прерывания. Затем, если прерывание разрешено, в счетчик команд загружается адрес вектора соответствующего прерывания. При этом флаг прерывания аппаратно сбрасывается. Он также может быть сброшен программно, путем записи лог. 1 в разряд регистра, соответствующий флагу.

Прерывания второго типа не имеют флагов прерываний и генерируются в течение всего времени, пока присутствуют условия, необходимые для генерации прерывания. Соответственно, если условия, вызывающие прерывание, 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 подпрограмму обработки прерывания (если это необходимо) и восстанавливать его значение перед вызовом команды RETI.

Микроконтроллеры семейства Mega поддерживают очередь прерываний, которая работает следующим образом: если условия генерации одного или более прерываний возникают в то время, когда флаг общего разрешения прерываний сброшен (все прерывания запрещены), соответствующие флаги устанавливаются в «1» и остаются в этом состоянии до установки флага общего разрешения прерываний. После разрешения прерываний выполняется их обработка в порядке приоритета.

Наименьшее время отклика для любого прерывания составляет 4 машинных цикла, в течение которых происходит сохранение счетчика команд в стеке. В течение последующих двух (для моделей ATmega8x и ATmega8515x) или трех циклов выполняется команда перехода к подпрограмме обработки прерывания. Если прерывание произойдет во время выполнения команды, длящейся несколько циклов, то генерация прерывания произойдет только после выполнения этой команды. Если же прерывание произойдет во время нахождения микроконтроллера в «спящем» режиме, время отклика увеличивается еще на 4 машинных цикла.

Возврат в основную программу занимает 4 машинных цикла, в течение которых происходит восстановление счетчика команд из стека. После выхода из прерывания процессор всегда выполняет одну команду основной программы, прежде чем обслужить любое отложенное прерывание.

11.4. Внешние прерывания

Для разрешения/запрещения внешних прерываний во всех моделях, кроме ATmega64x и ATmega128x, предназначен регистр GICR (GIMSK в моделях ATmega161x и ATmega163x), расположенный по адресу $3B ($5B). Формат этого регистра для различных моделей показан на **Рис. 2.54**, а описание его разрядов, относящихся к прерываниям, приведено в **Табл. 2.63**.

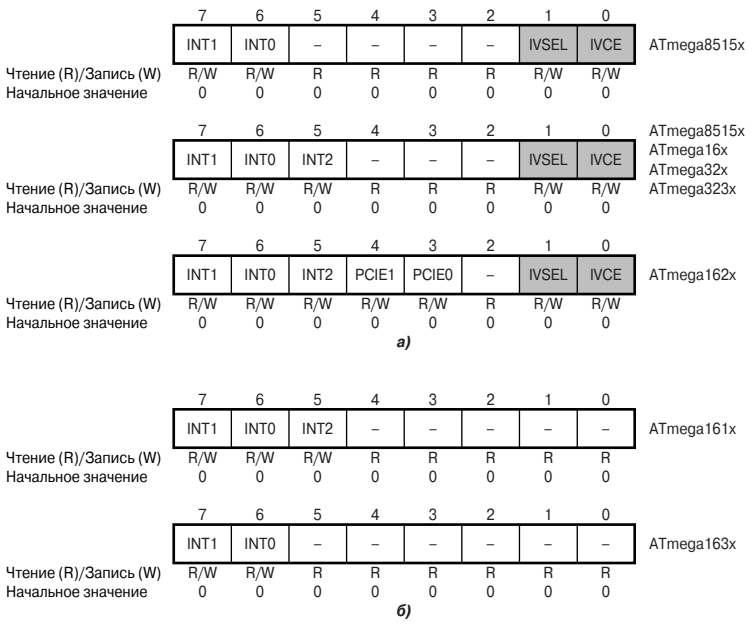


Рис. 2.54. Формат регистров GICR *(а)* и GIMSK *(б)*

Таблица 2.63. **Разряды регистров GICR (GIMSK)**

Разряд	Название	Описание	Модель
7	INT1	**Разрешение внешнего прерывания INT1.** Если в этом разряде записана лог. 1 и флаг I регистра SREG также установлен в «1», то разрешается внешнее прерывание с вывода INT1. Условие генерации прерывания определяется содержимым разрядов ISC01 и ISC00 регистра MCUCR	Все модели
6	INT0	**Разрешение внешнего прерывания INT0.** Если в этом разряде записана лог. 1 и флаг I регистра SREG также установлен в «1», то разрешается внешнее прерывание с вывода INT1. Условие генерации прерывания определяется содержимым разрядов ISC01 и ISC00 регистра MCUCR	Все модели
5	INT2	**Разрешение внешнего прерывания INT2.** Если в этом разряде записана лог. 1 и флаг I регистра SREG также установлен в «1», то разрешается внешнее прерывание с вывода INT2. Условие генерации прерывания определяется содержимым разряда ISC2	ATmega8515x ATmega16x ATmega161x ATmega162x ATmega32x ATmega323x

Продолжение таблицы 2.63

Разряд	Название	Описание	Модель
4	PCIE1	**Разрешение прерывания по изменению состояния выводов 1-й группы.** Если в этом разряде записана лог. 1 и флаг I регистра SREG также установлен в «1», то разрешается внешнее прерывание по изменению состояния выводов PCINT15...8 микроконтроллера. К возникновению прерывания приводит любое изменение сигнала на любом выводе	ATmega162x
3	PCIE0	**Разрешение прерывания по изменению состояния выводов 0-й группы.** Если в этом разряде записана лог. 1 и флаг I регистра SREG также установлен в «1», то разрешается внешнее прерывание по изменению состояния выводов PCINT7...0 микроконтроллера. К возникновению прерывания приводит любое изменение сигнала на любом выводе	ATmega162x

Какие именно из выводов PCINT15...0 могут влиять на генерацию прерываний PCINT1 и PCINT0, определяется регистрами PCMSK0 (выводы PCINT7...0) и PCMSK1 (выводы PCINT15...8), расположенными в пространстве дополнительных РВВ по адресам ($6B) и ($6C) соответственно. Формат этих регистров приведен на **Рис. 2.55**. Если какой-либо разряд этих регистров установлен в «1», то изменение состояния соответствующего вывода может вызвать прерывание.

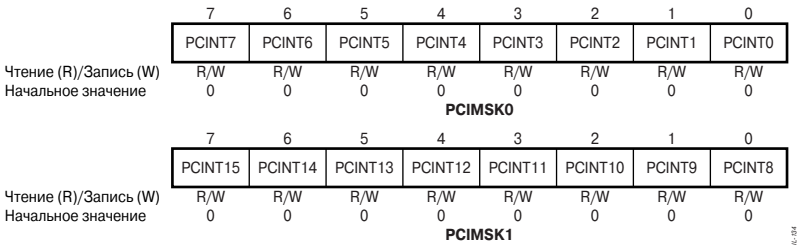


Рис. 2.55. Формат регистров PCMSK0 и PCMSK1

Напоминаем, что прерывания PCINT0 и PCINT1 доступны только в режиме совместимости с микроконтроллером ATmega161x (конфигурационная ячейка M161C = «0»).

Регистр GIFR (General Interrupt Flag Register — общий регистр флагов прерываний), расположенный по адресу $3A ($5A), предназначен для индикации наступления внешних прерываний в этих моделях. Формат этого регистра показан на **Рис. 2.56**, а описание его разрядов приведено в **Табл. 2.64**.

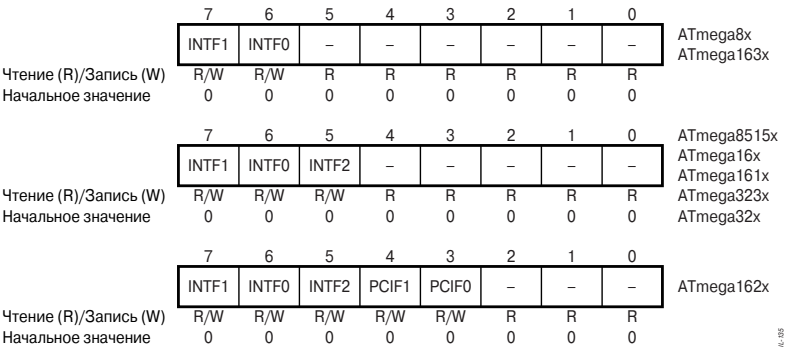


Рис. 2.56. Формат регистра GIFR

Таблица 2.64. **Разряды регистров GIFR**

Разряд	Название	Описание	Модель
7	INTF1	**Флаг внешнего прерывания INT1.** Если в результате события на выводе INT1 сформировался запрос на внешнее прерывание, этот разряд устанавливается в «1». Флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания или программно, записью в него лог. 1. Флаг INTF1 сброшен постоянно, если генерация прерывания должна происходить по НИЗКОМУ уровню на выводе INT1	Все модели
6	INTF0	**Флаг внешнего прерывания INT0.** Если в результате события на выводе INT0 сформировался запрос на внешнее прерывание, этот разряд устанавливается в «1». Флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания или программно, записью в него лог. 1. Флаг INTF0 сброшен постоянно, если генерация прерывания должна происходить по НИЗКОМУ уровню на выводе INT0	Все модели
5	INTF2	**Флаг внешнего прерывания INT2.** Если в результате события на выводе INT2 сформировался запрос на внешнее прерывание, этот разряд устанавливается в «1». Флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания или программно, записью в него лог. 1. Флаг INTF2 сброшен постоянно, если генерация прерывания должна происходить по НИЗКОМУ уровню на выводе INT2	ATmega8515x ATmega16x ATmega161x ATmega162x ATmega323x
4	PCIF1	**Разрешение прерывания по изменению состояния выводов 1-й группы.** Если в результате события на любом из выводов PCINT15...8 сформировался запрос на прерывание, этот разряд устанавливается в «1». Флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания или программно, записью в него лог. 1	ATmega162x

Продолжение таблицы 2.64

Разряд	Название	Описание	Модель
3	PCIF0	**Разрешение прерывания по изменению состояния выводов 0-й группы.** Если в результате события на любом из выводов PCINT7...0 сформировался запрос на прерывание, этот разряд устанавливается в «1». Флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания или программно, записью в него лог. 1	ATmega162x

Прерывания INT0 и INT1 могут быть сгенерированы по нарастающему/спадающему фронту сигнала или при появлении НИЗКОГО уровня на входе. Условия генерации этих прерываний определяются состоянием младших 4-х разрядов регистра MCUCR (**Рис. 2.57**) согласно **Табл. 2.65**.

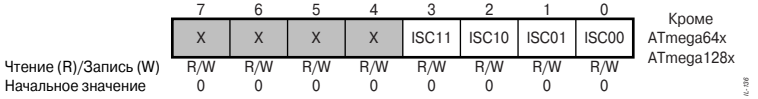


Рис. 2.57. Формат регистра MCUCR

Таблица 2.65. **Определение условий генерации внешних прерываний (разряды регистра MCUCR)**

Разряд	Название	Описание		
3, 2	ISC11, ISC10	Определяют условие генерации внешнего прерывания INT1 следующим образом:		
		ISC11	ISC10	Условие
		0	0	По НИЗКОМУ уровню на выводе INT1
		0	1	Зарезервировано
		1	0	По спадающему фронту сигнала на выводе INT1
		1	1	По нарастающему фронту сигнала на выводе INT1
1, 0	ISC01, ISC00	Определяют условие генерации внешнего прерывания INT0 следующим образом:		
		ISC01	ISC00	Условие
		0	0	По НИЗКОМУ уровню на выводе INT0
		0	1	Зарезервировано
		1	0	По спадающему фронту сигнала на выводе INT0
		1	1	По нарастающему фронту сигнала на выводе INT0

Обнаружение фронтов сигналов на выводах INT0 и INT1 осуществляется синхронно, поэтому минимальная длительность импульса, гарантирующая генерацию прерывания, составляет один период тактового сигнала микроконтроллера. Если генерация прерывания должна происходить по НИЗКОМУ уровню, то он должен удерживаться на выводе до окончания выполнения текущей команды, в противном случае прерывание сгенерировано не будет.

Прерывание INT2 может быть сгенерировано только по нарастающему или спадающему фронту сигнала. Условие генерации прерывания INT2 определяется состоянием разряда ISC2 (0-й разряд регистра EMCUCR в моделях ATmega8515x, ATmega161x, ATmega163x и 6-й разряд регистра MCUCSR в моделях ATmega16x, ATmega32x и ATmega323x). Если этот разряд сброшен в «0» (начальное состояние), прерывание будет сгенерировано по спадающему фронту сигнала, если установлен в «1» — по нарастающему фронту. Обнаружение перепадов сигнала на выводе INT2 осуществляется асинхронно, при этом минимальная длительность импульса, гарантирующая генерацию прерывания, составляет 50 нс.

В моделях ATmega64x и ATmega128x для разрешения/запрещения внешних прерываний предназначен регистр EIMSK (External Interrupt MaSK — регистр маски внешних прерываний), расположенный по адресу $39 ($59). Формат этого регистра показан на **Рис. 2.58**.

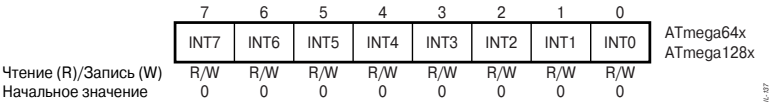


Рис. 2.58. Формат регистра EIMSK

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

Условия генерации прерываний определяются регистрами EICRA («A»-регистр управления внешними прерываниями) и EICRB («B»-регистр управления внешними прерываниями), расположенных по адресам $6A и $3A ($5A) соответственно. Формат этих регистров показан на **Рис. 2.59**, а описание их разрядов приведено в **Табл. 2.66**.

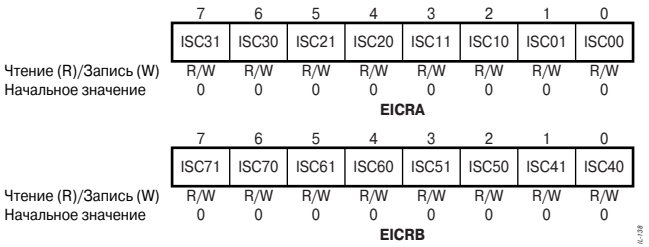


Рис. 2.59. Формат регистров EICRA и EICRB

Таблица 2.66. **Определение условий генерации внешних прерываний (разряды регистров EICRA и EICRB)**

Регистр	Разряд	Название	Описание		
EICRA	7, 6	ISC31, ISC30	Определяют условие генерации внешних прерываний INT7...INT0 следующим образом:		
	5, 4	ISC21, ISC20			
	3, 2	ISC11, ISC10			
	1, 0	ISC01, ISC00	ISC*n*1	ISC*n*0	Условие
EICRB	7, 6	ISC71, ISC70	0	0	По НИЗКОМУ уровню на выводе INT*n*
	5, 4	ISC61, ISC60	0	1	Зарезервировано
	3, 2	ISC51, ISC50	1	0	По спадающему фронту сигнала на выводе INT*n*
	1, 0	ISC41, ISC40	0	1	По нарастающему фронту сигнала на выводе INT*n*

Обнаружение фронтов сигналов на выводах INT3...INT0 осуществляется асинхронно, при этом минимальная длительность импульса, гарантирующая генерацию прерывания, составляет 50 нс. А на выводах INT7...INT4 обнаружение фронтов сигналов осуществляется синхронно, поэтому минимальная длительность импульса, гарантирующая генерацию прерывания, составляет 1 период тактового сигнала микроконтроллера. Если генерация прерывания должна происходить по НИЗКОМУ уровню, то он должен удерживаться на выводе до окончания выполнения текущей команды, в противном случае генерации прерывания не произойдет.

Для индикации возникновения внешних прерываний в моделях ATmega64x и ATmega128x предназначен регистр EIFR (External Interrupt Flag Register — регистр флагов внешних прерываний), расположенный по адресу $38 ($58). Формат этого регистра показан на **Рис. 2.60**.

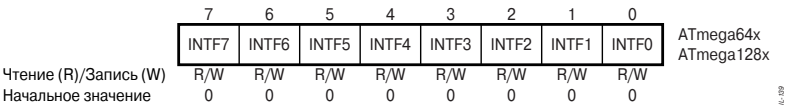


Рис. 2.60. Формат регистра EIFR

При возникновении запроса на прерывание на выводе INT*n* соответствующий флаг INTF*n* (*n*-й разряд регистра) устанавливается в «1». Флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания или программно, записью в него лог. 1. Здесь следует иметь в виду, что для программного сброса одного флага нельзя использовать команду SBI (установить разряд регистра ввода/вывода), т. к. при ее выполнении будут сброшены все остальные флаги. Это связано с тем, что команда сначала считывает содержимое регистра, затем изменяет указанный разряд и записывает полученный результат обратно в регистр. Аналогично, при выполнении команды CBI (сбросить разряд регистра ввода/вывода) будут сброшены все разряды регистра, кроме указанного. Если генерация прерывания должна происходить по НИЗКОМУ уровню на выводе, соответствующий флаг сброшен постоянно.

В заключение скажем, что все внешние прерывания генерируются даже в том случае, если соответствующие выводы сконфигурированы как выходы. Эта особенность микроконтроллеров позволяет генерировать прерывания программно.

Глава 12. **Порты ввода/вывода**

12.1. Общие сведения

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

Выходные буферы всех портов, имея симметричные нагрузочные характеристики, обеспечивают высокую нагрузочную способность при любом уровне сигнала. Нагрузочной способности достаточно для непосредственного 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 V_{CC}.

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

Микроконтроллеры различных моделей семейства имеют различное количество портов и соответственно контактов ввода/вывода:

- ATmega8x имеют три порта ввода/вывода: порт B (8-разрядный), порт C (7-разрядный) и порт D (8-разрядный). Всего контактов ввода/вывода 23;
- ATmega8515x имеют четыре 8-разрядных порта ввода/вывода (порты A, B, C, D) и один 3-разрядный порт ввода/вывода E. Всего кон-

тактов ввода/вывода 35. В режиме совместимости с микроконтроллерами AT90S4414/8515 семейства Classic выводы порта E используются только соответствующими периферийными устройствами;
- ATmega16x имеют четыре 8-разрядных порта ввода/вывода (порты A, B, C, D). Всего контактов ввода/вывода 32;
- ATmega161x имеют четыре 8-разрядных порта ввода/вывода (порты A, B, C, D) и один 3-разрядный порт ввода/вывода E. Всего контактов ввода/вывода 35;
- ATmega162x также имеют четыре 8-разрядных порта ввода/вывода (порты A, B, C, D) и один 3-разрядный порт ввода/вывода E. всего контактов ввода/вывода 35;
- ATmega163x, ATmega32x и ATmega323x имеют четыре 8-разрядных порта ввода/вывода (порты A, B, C, D). Всего контактов ввода/вывода 32;
- ATmega64x и ATmega128x имеют шесть 8-разрядных порта ввода/вывода (порты A, B, C, D, E, F) и один 5-разрядный порт ввода/вывода G. Всего контактов ввода/вывода 53.

12.2. 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 порта PORT*x*, регистр направления данных DDR*x* и регистр выводов порта PIN*x*. Действительные названия регистров получаются подстановкой названия порта вместо символа «*x*», соответственно регистры порта A называются PORTA, DDRA, PINA, порта B — PORTB, DDRB, PINB и т. д. Поскольку с помощью регистров PIN*x* 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.

Адреса регистров всех портов ввода/вывода приведены в **Табл. 2.67**.

Таблица 2.67. **Регистры портов ввода/вывода**

Порт	Регистр	ATmega8x	ATmega8515x	ATmega16x	ATmega161x ATmega162x	ATmega163x ATmega323x ATmega32x	ATmega64x ATmega128x
A	PORTA	—	$1B ($3B)	$1B ($3B)	$1B ($3B)	$1B ($3B)	$1B ($3B)
	DDRA	—	$1A ($3A)	$1A ($3A)	$1A ($3A)	$1A ($3A)	$1A ($3A)
	PINA	—	$19 ($39)	$19 ($39)	$19 ($39)	$19 ($39)	$19 ($39)

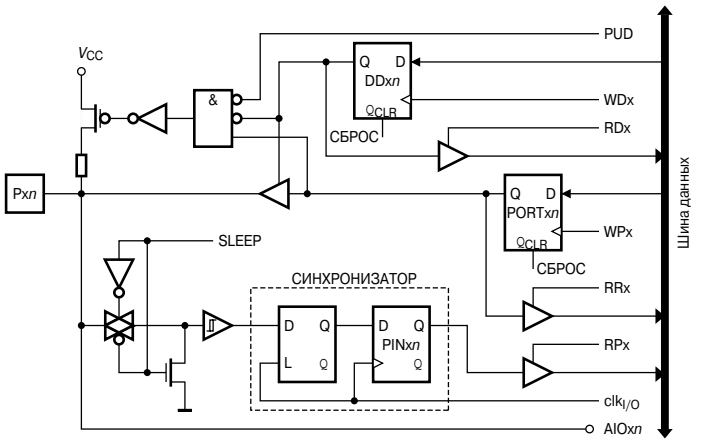
Продолжение таблицы 2.67

Порт	Регистр	ATmega8x	ATmega8515x	ATmega16x	ATmega161x ATmega162x	ATmega163x ATmega323x ATmega32x	ATmega64x ATmega128x
B	PORTB	$18 ($38)	$18 ($38)	$18 ($38)	$18 ($38)	$18 ($38)	$18 ($38)
	DDRB	$17 ($37)	$17 ($37)	$17 ($37)	$17 ($37)	$17 ($37)	$17 ($37)
	PINB	$16 ($36)	$16 ($36)	$16 ($36)	$16 ($36)	$16 ($36)	$16 ($36)
C	PORTC	$15 ($38)	$15 ($38)	$15 ($38)	$15 ($38)	$15 ($38)	$15 ($38)
	DDRC	$14 ($37)	$14 ($37)	$14 ($37)	$14 ($37)	$14 ($37)	$14 ($37)
	PINC	$13 ($36)	$13 ($36)	$13 ($36)	$13 ($36)	$13 ($36)	$13 ($36)
D	PORTD	$12 ($32)	$12 ($32)	$12 ($32)	$12 ($32)	$12 ($32)	$12 ($32)
	DDRD	$11 ($31)	$11 ($31)	$11 ($31)	$11 ($31)	$11 ($31)	$11 ($31)
	PIND	$10 ($30)	$10 ($30)	$10 ($30)	$10 ($30)	$10 ($30)	$10 ($30)
E	PORTE	—	$07 ($27)	—	$07 ($27)	—	$03 ($23)
	DDRE	—	$06 ($26)	—	$06 ($26)	—	$02 ($22)
	PINE	—	$05 ($25)	—	$05 ($25)	—	$01 ($21)
F	PORTF	—	$03 ($23)	—	—	—	($62)
	DDRF	—	$02 ($22)	—	—	—	($61)
	PINF	—	$01 ($21)	—	—	—	$00 ($20)
G	PORTG	—	—	—	—	—	($65)
	DDRG	—	—	—	—	—	($64)
	PING	—	—	—	—	—	($63)

12.3. Конфигурирование портов ввода/вывода

Структурная схема одного из каналов порта ввода/вывода P*xn* при работе его в качестве цифрового входа/выхода общего назначения приведена на **Рис. 2.61**.

Каждому выводу порта соответствуют три разряда регистров ввода/вывода: PORT*xn* (регистр PORT*x*), DD*xn* (регистр DDR*x*) и PIN*xn* (регистр PIN*x*). Действительные названия разрядов регистров получаются подстановкой названия порта вместо символа «*x*» и номера разряда вместо символа «*n*». Порядковый номер вывода порта соответствует порядковому номеру разряда регистров этого порта. Поэтому, если разрядность порта меньше восьми, в регистрах порта используется соответствующее число младших разрядов. Незадействованные старшие разряды регистров доступны только для чтения и всегда содержат «0».



PUD:	выключение подт. резистора	WDx:	запись регистра DDRx
SLEEP:	управление спящим режимом	RDx:	чтение регистра DDRx
$clk_{I/O}$:	тактовый сигнал подсистемы ввода/вывода	WPx:	запись регистра PORTx
		RRx:	чтение регистра PORTx
		RPx:	чтение выводов порта PORTx
		AIO*xn*:	аналоговый вход/выход вывода *n* порта PORTx

Примечание. Сигналы WPx, WDx, RPx, RDx являются общими для всех выводов одного порта; сигналы $clk_{I/O}$, SLEEP и PUD являются общими для всех портов микроконтроллера.

Рис. 2.61. Структурная схема контакта ввода/вывода

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

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

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

Вообще говоря, управление подтягивающими резисторами почти во всех микроконтроллерах семейства, за исключением моделей ATmega161x, осуществляется на двух уровнях. Общее управление (для всех выводов портов) осуществляется разрядом PUD (2-й разряд) регистра специальных функций SFIOR. В моделях ATmega64x и ATmega128x этот регистр располагается по адресу $20 ($40), а в остальных моделях — по адресу $30 ($50).

Если разряд PUD сброшен в «0» (начальное состояние), состояние подтягивающих резисторов будет определяться состоянием разрядов PORT*xn* для каждого входа порта. Если же разряд PUD установлен в «1», подтягивающие резисторы отключаются от всех выводов микроконтроллера.

Обратите внимание, что при переключении вывода между третьим состоянием (DD*xn* = «0», PORT*xn* = «0») и состоянием ВЫСОКОГО уровня (DD*xn* = «1», PORT*xn* = «1») происходит переход через одно из промежуточных состояний: либо включается подтягивающий резистор (DD*xn* = «0», PORT*xn* = «1»), либо выход переключается в состояние НИЗКОГО уровня (DD*xn* = «1», PORT*xn* = «0»). Наиболее применимым является, как правило, первый вариант, поскольку для высокоимпедансных систем безразлично, каким образом формируется ВЫСОКИЙ уровень. Если в каком-либо случае это не подходит, пользователь может отключить подтягивающие резисторы от всех портов установкой в «1» разряда PUD регистра SFIOR.

Аналогичная ситуация возникает и при переключении между состоянием с включенным подтягивающим резистором (DD*xn* = «0», PORT*xn* = «1») и состоянием НИЗКОГО уровня (DD*xn* = «1», PORT*xn* = «0»). В этом случае промежуточным состоянием является либо высокоимпедансное состояние (DD*xn* = «0», PORT*xn* = «0»), либо состояние ВЫСОКОГО уровня (DD*xn* = «1», PORT*xn* = «1»).

Все возможные сочетания состояний управляющих разрядов и соответственно конфигурации выводов портов сведены в **Табл. 2.68**.

Таблица 2.68. **Конфигурации выводов портов**

DD*xn*	PORT*xn*	PUD* (в SFIOR)	Функция вывода	Резистор	Примечания
0	0	X	Вход	Отключен	Третье состояние (Hi-Z)**
0	1	0	Вход	Подключен	При подключении нагрузки между выводом и общим проводом, вывод является источником 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»

* Отсутствует в моделях ATmega161x.
** Состояние выводов порта при сбросе.

Состояние вывода микроконтроллера (независимо от установок разряда DD*xn*) может быть получено путем чтения разряда PIN*xn* регистра PIN*x*. При этом следует помнить, что между действительным изменением сигнала на выводе и изменением разряда PIN*xn* существует задержка. Эта задержка вносится узлом синхронизации, состоя-

щего, как показано на **Рис. 2.61**, из разряда PIN*xn* и дополнительного триггера-защелки. Значение сигнала на выводе микроконтроллера фиксируется триггером-защелкой при НИЗКОМ уровне тактового сигнала и переписывается затем в разряд PIN*xn* по нарастающему фронту тактового сигнала. Соответственно величина задержки может составлять от 0.5 до 1.5 периода системного тактового сигнала, как показано на **Рис. 2.62, 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 положительному фронту тактового сигнала, задержка в этом случае равна одному периоду тактового сигнала (**Рис. 2.62, б**).

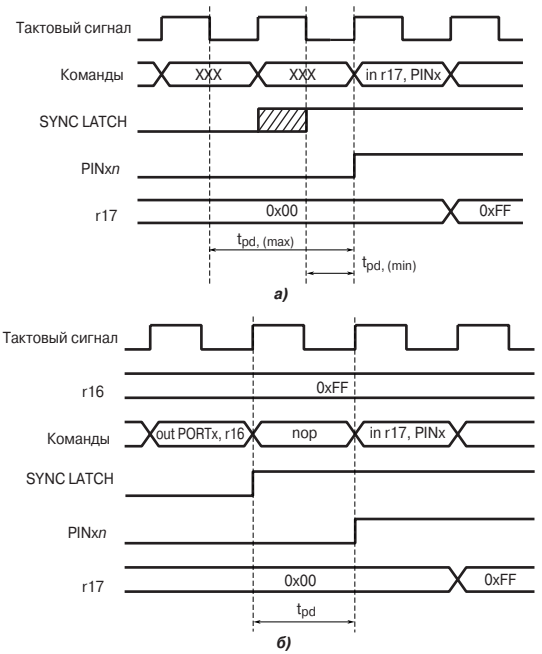


Рис. 2.62. Синхронизация при чтении состояния вывода: *а* — при считывании состояния разряда PINxn; *б* — при повторном считывании

Далее приведен пример конфигурирования одного из портов микроконтроллера. В примере выводы 0 и 1 порта B устанавливаются в «1», выводы 2 и 3 — в «1». Выводы 4...7 порта конфигурируются как входы, при этом к выводам 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
                            ;подтягивающих резисторов
out DDRB,r17                ;Задать режимы работы выводов
nop                         ;для синхронизации
in r16,PINB                 ;Считать состояние выводов порта
...
```

Пример на C

```
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 устройствами микроконтроллеров. При этом возможны две ситуации. В одних случаях пользователь должен самостоятельно задавать конфигурацию вывода, а в других — вывод конфигурируется автоматически при включении соответствующего периферийного устройства. Об этом будет сказано при рассмотрении соответствующих периферийных устройств.

Глава 13. **Таймеры**

13.1. Общие сведения

Микроконтроллеры семейства в зависимости от модели имеют в своем составе от двух до четырех таймеров/счетчиков общего назначения (**Табл. 2.69**).

Таблица 2.69. **Таймеры/счетчики общего назначения**

Таймер/счетчик	**ATmega8x**	**ATmega8515x**	**ATmega16x**	**ATmega161x**	**ATmega162x**	**ATmega163x**	**ATmega32x**	**ATmega323x**	**ATmega64x**	**ATmega128x**
Таймер/счетчик T0 (8-разрядный)	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
Таймер/счетчик T1 (16-разрядный)	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
Таймер/счетчик T2 (8-разрядный)	♦	–	♦	♦	♦	♦	♦	♦	♦	♦
Таймер/счетчик T3 (16-разрядный)	–	–	–	–	♦	–	–	–	♦	♦

Как видно из таблицы, во всех моделях микроконтроллеров семейства присутствуют, как минимум, два таймера/счетчика — T0 и T1. Таймер/счетчик 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 возможность генерации сигналов с широтно-импульсной модуляцией (ШИМ) фиксированной разрядности, а также возможность работать в асинхронном режиме в качестве часов реального времени.

Таймер/счетчик T1 тоже может использоваться для отсчета временных интервалов и как счетчик внешних событий. Кроме того, он может выполнять запоминание своего состояния по внешнему сигналу. Как и таймер/счетчик T0, он может работать в качестве широтно-импульсного модулятора, но уже переменной разрядности и к тому же многоканального (количество каналов зависит от модели).

Таймер/счетчик T2 полностью аналогичен таймеру/счетчику T0. При наличии в микроконтроллере обоих таймеров/счетчиков, один из них может работать в асинхронном режиме, а другой — в качестве счетчика внешних событий.

Таймер/счетчик T3 по функциональным возможностям идентичен таймеру/счетчику T1.

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

13.2. Назначение выводов таймеров/счетчиков

Каждый таймер/счетчик использует один или более выводов микроконтроллера. Как правило, эти выводы — линии портов ввода/вывода общего назначения, а функции, реализуемые этими выводами при работе совместно с таймерами/счетчиками, являются их альтернативными функциями.

Все выводы микроконтроллеров, используемые таймерами/счетчиками общего назначения, сведены в **Табл. 2.70**. Там же указаны функции этих выводов.

Таблица 2.70. **Выводы, используемые таймерами/счетчиками общего назначения**

Название	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega32x	ATmega323x	ATmega64x	ATmega128x	Описание
T0	PD4	PB0	PB0	PB0	PB0	PB0	PB0	PB0	—	—	Вход внешнего сигнала таймера T0
OC0	—	PB0	PB3	PB0	PB0	—	PB3	PB3	PB4	PB4	Выход схемы сравнения таймера T0
T1	PD5	PB1	PB1	PB1	PB1	PB1	PB1	PB1	PD6	PD6	Вход внешнего сигнала таймера T1
ICP	PB0	PE0*	PD6	PE0	—	PD6	PD6	PD6	—	—	Вход захвата таймера T1
ICP1	—	—	—	—	PE0	—	—	—	PD4	PD4	

Продолжение таблицы 2.70

Название	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega32x	ATmega323x	ATmega64x	ATmega128x	Описание
OC1A	PB1	PD5	PD5	PD5	PD5	PD5	PD5	PD5	PB5	PB5	Выход схемы сравнения таймера T1
OC1B	PB2	PE2*	PD4	PE2	PE2	PD4	PD4	PD4	PB6	PB6	
OC1C	—	—	—	—	—	—	—	—	PB7	PB7	
T2	—	—	—	—	—	—	—	—	PD7	PD7	Вход внешнего сигнала таймера T2
OC2	PB3	—	PD7	PB1	PB1	PD7	PD7	PD7	PB7	PB7	Выход схемы сравнения таймера T2
T3	—	—	—	—	—	—	—	—	PE6	PE6	Вход внешнего сигнала таймера T3
ICP3	—	—	—	—	PD3	—	—	—	PE7	PE7	Вход захвата таймера T3
OC3A	—	—	—	—	PD4	—	—	—	PE3	PE3	Выход схемы сравнения таймера T3
OC3B	—	—	—	—	PB4	—	—	—	PE4	PE4	
OC3C	—	—	—	—	—	—	—	—	PE5	PE5	
TOSC1	PB6	—	PC6	PD4	PD4	PC6	PC6	PC6	PG4	PG4	Вход для подключения резонатора
TOSC2	PB7	—	PC7	PD5	PD5	PC7	PC7	PC7	PG3	PG3	Выход для подключения резонатора

* В режиме совместимости с микроконтроллерами AT90S4414/8515 — не контакт порта ввода/вывода, а выделенный вывод.

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.

13.3. Прерывания от таймеров/счетчиков

Для разрешения/запрещения прерываний от таймеров/счетчиков T0, T1 и T2 во всех моделях предназначен регистр TIMSK (Timer/Counter Interrupt MaSK Register — регистр маски прерываний от таймеров/счетчиков). В моделях ATmega64x и ATmega128x этот регистр расположен по адресу $37 ($57), а в остальных моделях — по адресу $39 ($59). Формат этого регистра для различных моделей показан на **Рис. 2.63**, а описание его разрядов приведено в **Табл. 2.71**.

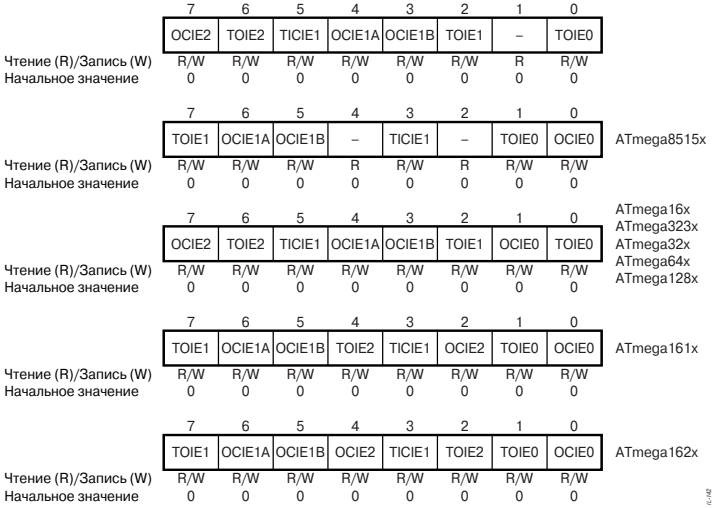


Рис. 2.63. Формат регистра TIMSK

Таблица 2.71. **Разряды регистра TIMSK**

Название	Описание
OCIE2	Флаг разрешения прерывания по событию «Совпадение» таймера/счетчика T2
TOIE2	Флаг разрешения прерывания по переполнению таймера/счетчика T2
TICIE1	Флаг разрешения прерывания по событию «Захват» таймера/счетчика T1
OCIE1A	Флаг разрешения прерывания по событию «Совпадение A» таймера/счетчика T1
OCIE1B	Флаг разрешения прерывания по событию «Совпадение B» таймера/счетчика T1
TOIE1	Флаг разрешения прерывания по переполнению таймера/счетчика T1
OCIE0	Флаг разрешения прерывания по событию «Совпадение» таймера/счетчика T0
TOIE0	Флаг разрешения прерывания по переполнению таймера /счетчика T0

Для разрешения/запрещения прерываний от таймеров/счетчиков T1 и T3 в моделях ATmega64x и ATmega128x имеется еще один регистр — ETIMSK (Enaible Timer/Counter Interrupt MaSK Register — регистр разрешения чтения маски прерываний от таймеров/счетчиков), расположенный по адресу ($7D) в пространстве дополнительных регистров ввода/вывода. Формат этого регистра приведен на **Рис. 2.64**, а описание его разрядов — в **Табл. 2.72**.

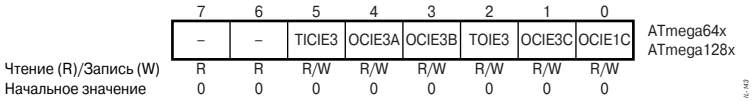


Рис. 2.64. Формат регистра ETIMSK

Таблица 2.72. **Разряды регистра ETIMSK**

Разряд	Название	Описание
7, 6	—	Не используется, читается как «0»
5	TICIE3	Флаг разрешения прерывания по событию «Захват» таймера/счетчика T3
4	OCIE3A	Флаг разрешения прерывания по событию «Совпадение A» таймера/счетчика T3
3	OCIE3B	Флаг разрешения прерывания по событию «Совпадение B» таймера/счетчика T3
2	TOIE3	Флаг разрешения прерывания по переполнению таймера/счетчика T3
1	OCIE3C	Флаг разрешения прерывания по событию «Совпадение C» таймера/счетчика T3
0	OCIE1C	Флаг разрешения прерывания по событию «Совпадение C» таймера/счетчика T1

Для разрешения какого-либо прерывания от таймера/счетчика необходимо установить в «1» соответствующий разряд регистра TIMSK (ETIMSK) и, разумеется, флаг I регистра SREG.

Для индикации наступления прерываний от таймеров T0, T1 и T2 предназначен регистр TIFR (Timer/Counter Interrupt Flag Register — регистр флагов прерываний от таймеров/счетчиков). В моделях ATmega64x и ATmega128x этот регистр расположен по адресу $36 ($56), а в остальных моделях — по адресу $38 ($58). Формат этого регистра для различных моделей показан на **Рис. 2.65**, а описание его разрядов приведено в **Табл. 2.73**.

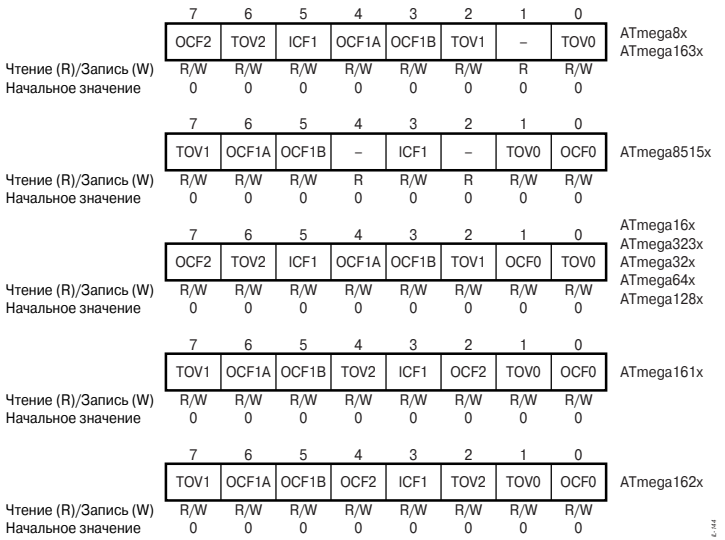


Рис. 2.65. Формат регистра TIFR

Таблица 2.73. **Разряды регистра TIFR**

Название	Описание
OCF2	Флаг прерывания по событию «Совпадение» таймера/счетчика T2
TOV2	Флаг прерывания по переполнению таймера/счетчика T2
ICF1	Флаг прерывания по событию «Захват» таймера/счетчика T1
OCF1A	Флаг прерывания по событию «Совпадение A» таймера/счетчика T1
OCF1B	Флаг прерывания по событию «Совпадение B» таймера/счетчика T1
TOV1	Флаг прерывания по переполнению таймера/счетчика T1
OCF0	Флаг прерывания по событию «Совпадение» таймера/счетчика T0
TOV0	Флаг прерывания по переполнению таймера /счетчика T0

Для разрешения/запрещения прерываний от таймеров/счетчиков T1 и T3 в моделях ATmega64x и ATmega128x предназначен регистр ETIFR, расположенный по адресу ($7C) в пространстве дополнительных регистров ввода/вывода. Формат этого регистра приведен на **Рис. 2.66**, а описание его разрядов — в **Табл. 2.74**.

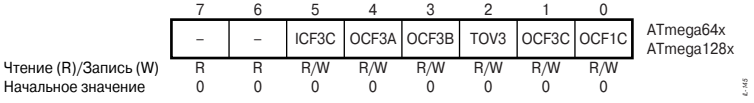


Рис. 2.66. Формат регистра ETIFR

Таблица 2.74. **Разряды регистра ETIFR**

Разряд	Название	Описание
7, 6	—	Не используется, читается как «0»
5	ICF3	Флаг прерывания по событию «Захват» таймера/счетчика T3
4	OCF3A	Флаг прерывания по событию «Совпадение A» таймера/счетчика T3
3	OCF3B	Флаг прерывания по событию «Совпадение B» таймера/счетчика T3
2	TOV3	Флаг прерывания по переполнению таймера/счетчика T3
1	OCF3C	Флаг прерывания по событию «Совпадение C» таймера/счетчика T3
0	OCF1C	Флаг прерывания по событию «Совпадение C» таймера/счетчика T1

При наступлении какого-либо события соответствующий флаг регистра TIFR (ETIFR) устанавливается в «1». При запуске подпрограммы обработки прерывания он аппаратно сбрасывается в «0». Любой флаг может быть сброшен в «0» также программно путем записи в него лог. 1.

13.4. 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. 2.67, а**. Структурная схема блока предделителя таймеров/счетчиков, имеющих возможность работы в асинхронном режиме, приведена на **Рис. 2.67, б**.

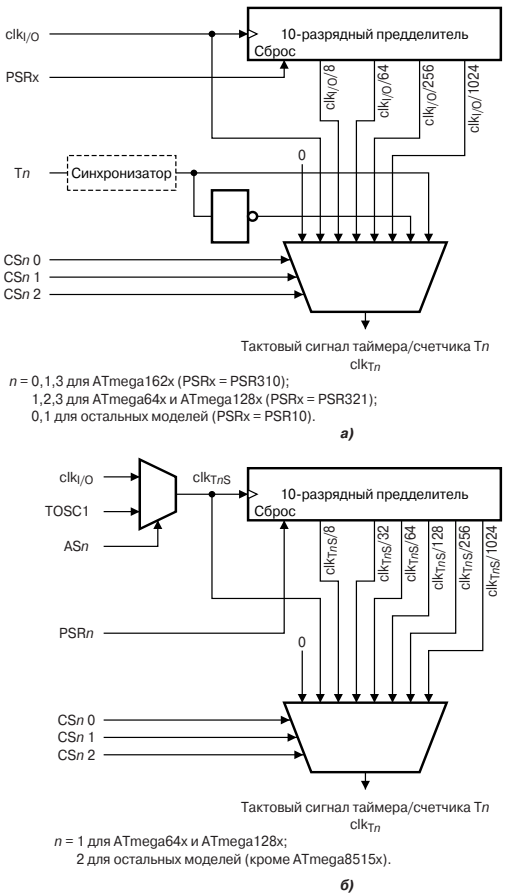


Рис. 2.67. Блок предделителя таймера/счетчика:
а — без асинхронного режима; *б* — с асинхронным режимом

Как показано на рисунке, в состав каждого блока входят собственно 10-разрядный предделитель и выходной мультиплексор (селектор тактового сигнала), а для таймеров, имеющих возможность работы в асинхронном режиме, — еще и входной мультиплексор исходного тактового сигнала. По схеме, приведенной на **Рис. 2.67**, выполнен предделитель таймера/счетчика T0 моделей ATmega64x/128x и таймера/счетчика T2 остальных моделей.

Следует иметь в виду, что все таймеры/счетчики каждой модели семейства, не имеющие асинхронного режима работы, используют один и тот же 10-разрядный предделитель. Соответственно в моделях ATmega64x и ATmega128x это таймеры/счетчики T1, T2 и T3; в моделях ATmega162x — T0, T1 и T3; в остальных моделях — T0 и T1. При этом управление тактовым сигналом каждого таймера/счетчика осуществляется индивидуально и будет описано при их рассмотрении.

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.

13.4.1. Управление предделителями

Помимо управления тактовым сигналом таймера/счетчика, все микроконтроллеры семейства позволяют осуществлять сброс предделителей, а модели ATmega162x, ATmega64x и ATmega128x позволяют также осуществлять их остановку. Для этого используется регистр специальных функций SFIOR. Формат этого регистра для различных моделей микроконтроллеров приведен на **Рис. 2.68** (разряды, не используемые для управления предделителями таймеров/счетчиков, указаны на рисунке как «X»).

Для сброса предделителей таймеров/счетчиков используются разряды PSR*x* регистра. При записи в эти разряды лог. 1 предделители соответствующих таймеров/счетчиков переводятся в исходное состояние. Разряды сбрасываются в «0» аппаратно после выполнения операции сброса. Еще раз напоминаем, что один предделитель может использоваться несколькими таймерами/счетчиками и, соответственно, сброс предделителя повлияет на все таймеры/счетчики, использующие его.

Остановка всех предделителей микроконтроллера осуществляется записью лог. 1 в разряд TSM регистра SFIOR. Последующий запуск предделителей осуществляется записью в разряд TSM лог. 0. Указанная функция может

использоваться, в частности, для синхронизации таймеров/счетчиков. После установки разряда TSM и требуемых разрядов PSR*x*, соответствующие таймеры/счетчики останавливаются и могут быть проинициализированы требуемыми значениями. После сброса разряда TSM все таймеры/счетчики начнут работать одновременно.

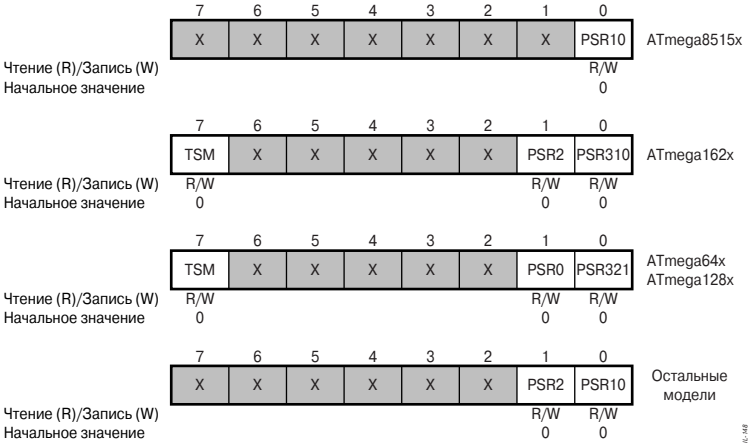


Рис. 2.68. Регистр SFIOR

13.4.2. Использование внешнего тактового сигнала

Таймеры/счетчики, в которых в качестве тактового может использоваться внешний сигнал, указаны в **Табл. 2.75**. Сразу отметим, что это таймеры/счетчики, не имеющие асинхронного режима работы.

Таблица 2.75. **Внешний тактовый сигнал в таймерах/счетчиках**

Таймер/счетчик	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega32x	ATmega323x	ATmega64x	ATmega128x
T0	♦	♦	♦	♦	♦	♦	♦	♦	—	—
T1	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
T2	—	—	—	—	—	—	—	—	♦	♦
T3	—	—	—	—	—	—	—	—	♦	♦

Внешний сигнал, поступающий на вход T*n* (*n* = 0...3) микроконтроллера, прежде чем поступить на вход селектора тактового сигнала проходит через специальный узел, включающий схему синхронизации и детектор фронта.

Синхронизация внешнего сигнала осуществляется с частотой тактового генератора микроконтроллера (состояние вывода T1 считывается по нарастающему фронту тактового сигнала $clk_{I/O}$). Поэтому частота внешнего сигнала должна быть в 2 раза меньше частоты тактового сигнала микроконтроллера ($f_{EXT} < f_{clk\_I/O}/2$). Однако, чтобы гарантировать обнаружение фронтов внешнего сигнала во всем диапазоне возможных изменений частоты и скважности тактового сигнала микроконтроллера (из-за разброса параметров элементов тактового генератора), частота внешнего сигнала должна быть еще меньше — $f_{clk\_I/O}/2.5$.

Также следует понимать, что входной каскад вносит задержку между изменением состояния вывода и обновлением счетного регистра таймера/счетчика. Величина задержки составляет 2.5...3.5 машинных цикла.

13.5. Таймеры/счетчики T0 и T2

Восьмиразрядный таймер/счетчик T0 присутствует во всех моделях микроконтроллеров семейства Mega, а таймер/счетчик T2 — во всех, кроме ATmega8515x. Всего в микроконтроллерах семейства реализовано три исполнения восьмиразрядных таймеров/счетчиков, отличающихся набором выполняемых функций. Структурные схемы всех трех вариантов приведены на **Рис. 2.69**, а функции, которые они могут выполнять, перечислены в **Табл. 2.76** и **Табл. 2.77**.

В состав таймеров/счетчиков первого исполнения (T0 в моделях ATmega8x и ATmega163x) входят два регистра ввода/вывода: счетный регистр TCNT0 и регистр управления TCCR0. В состав таймеров/счетчиков второго исполнения (T0 в моделях ATmega8515x, ATmega16x, ATmega161x, ATmega162x, ATmega32x, ATmega323x и T2 в моделях ATmega64x и ATmega128x) входят уже 3 регистра ввода/вывода: счетный регистр TCNT0 (TCNT2), регистр управления TCCR0 (TCCR2) и регистр сравнения OCR0 (OCR2). В третьем исполнении (T0 в моделях ATmega64x/128x и T2 в остальных моделях) добавляется регистр ASSR, служащий для управления модулем таймера/счетчика в асинхронном режиме. Адреса всех перечисленных регистров указаны в **Табл. 2.78**.

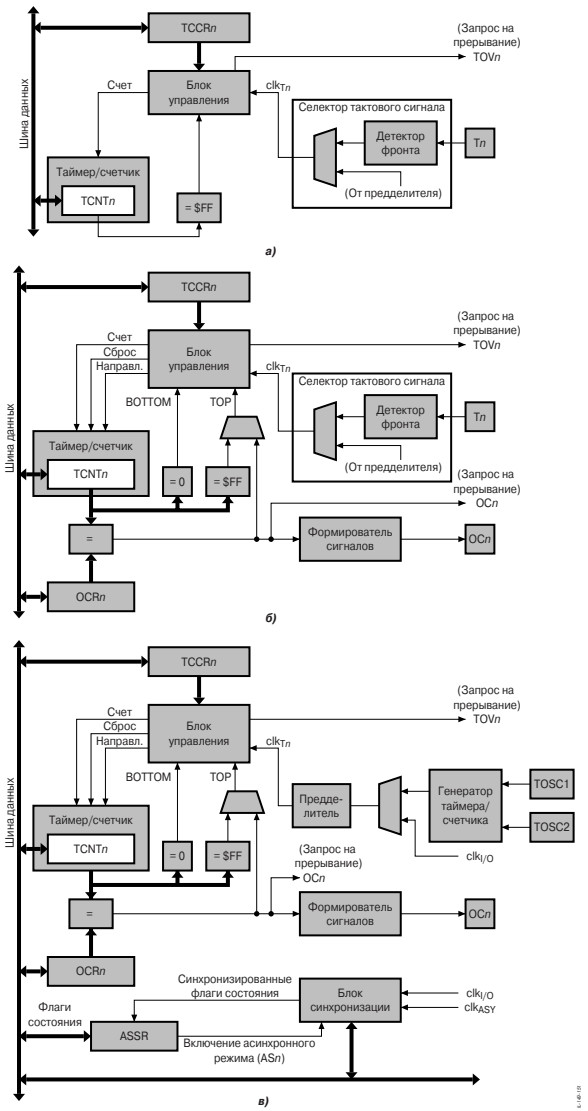


Рис. 2.69. Структурные схемы 8-разрядных таймеров/счетчиков (T0 и T2): *а* — исполнение 1; *б* — исполнение 2; *в* — исполнение 3

Таблица 2.76. **Функции таймера/счетчика T0**

Модель	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega32x	ATmega323x	ATmega64x	ATmega128x
Вариант исполнения	1	2	2	2	2	1	2	2	3	3
Восьмиразрядный счетчик	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
Счетчик внешних событий	♦	♦	♦	♦	♦	♦	♦	♦	—	—
Широтно-импульсный модулятор (8-разрядный)	—	♦	♦	♦	♦	—	♦	♦	♦	♦
Формирователь сигналов	—	♦	♦	♦	♦	—	♦	♦	♦	♦
Часы реального времени	—	—	—	—	—	—	—	—	♦	♦

Таблица 2.77. **Функции таймера/счетчика T2**

Модель	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega32x	ATmega323x	ATmega64x	ATmega128x
Вариант исполнения	3	—	3	3	3	3	3	3	2	2
Восьмиразрядный счетчик	♦	—	♦	♦	♦	♦	♦	♦	♦	♦
Счетчик внешних событий	—	—	—	—	—	—	—	—	♦	♦
Широтно-импульсный модулятор (8-разрядный)	♦	—	♦	♦	♦	♦	♦	♦	♦	♦
Формирователь сигналов	♦	—	♦	♦	♦	♦	♦	♦	♦	♦
Часы реального времени	♦	—	♦	♦	♦	♦	♦	♦	—	—

Количество прерываний, которые могут генерировать таймеры/счетчики T0 и T2, также зависит от исполнения. Таймер/счетчик T0 первого исполнения может генерировать прерывание только при переполнении счетного регистра. В таймерах/счетчиках второго и третьего исполнения прерывание может генерироваться также при равенстве счетного регистра и регистра сравнения. Флаги обоих прерываний находятся в регистре TIFR, а разрешение/запрещение этих прерываний осуществляется установкой/сбросом соответствующих флагов регистра TIMSK.

Таблица 2.78. **Регистры 8-разрядных таймеров/счетчиков**

Регистр	Адрес	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega323x	ATmega32x	ATmega64x	ATmega128x
TCCR0	$33 ($53)	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
TCNT0	$32 ($52)	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
OCR0	$31 ($51)	–	♦	–	♦	♦	–	–	–	♦	♦
	$3C ($5C)	–	–	♦	–	–	–	♦	♦	–	–
TCCR2	$25 ($45)	♦	–	♦	–	–	♦	♦	♦	♦	♦
	$27 ($47)	–	–	–	♦	♦	–	–	–	–	–
TCNT2	$24 ($44)	♦	–	♦	–	–	♦	♦	♦	♦	♦
	$23 ($43)	–	–	–	♦	♦	–	–	–	–	–
OCR2	$23 ($43)	♦	–	♦	–	–	♦	♦	♦	♦	♦
ASSR	$22 ($42)	♦	–	♦	–	–	♦	♦	♦	–	–
	$26 ($46)	–	–	–	♦	♦	–	–	–	–	–
	$30 ($50)	–	–	–	–	–	–	–	–	♦	♦

Счетный регистр таймера/счетчика TCNT0 (TCNT2) входит в состав основного блока модуля — блока реверсивного счетчика. В зависимости от режима работы модуля содержимое счетного регистра сбрасывается, инкрементируется или декрементируется по каждому импульсу тактового сигнала таймера/счетчика clk_{T0} (clk_{T2}). Независимо от того, присутствует тактовый сигнал или нет, регистр доступен в любой момент времени как для чтения, так и для записи. Однако следует помнить, что любая операция записи в счетный регистр блокирует работу блока сравнения на время одного периода тактового сигнала таймера/счетчика. После подачи напряжения питания в регистре TCNT0 (TCNT2) находится нулевое значение. При некоторых изменениях состояния таймера/счетчика, определяемых режимом его работы, устанавливается флаг TOV0 (TOV2) регистра TIFR. Разрешение

прерывания осуществляется установкой в «1» разряда TOIE0 (TOIE2) регистра TIMSK (разумеется, флаг I регистра SREG также должен быть установлен в «1»).

Регистр сравнения OCR0 (OCR2) входит в состав блока сравнения модуля. Во время работы таймера/счетчика производится непрерывное (в каждом машинном цикле) сравнение этого регистра с регистром TCNT0 (TCNT2). В случае равенства содержимого этих регистров в следующем машинном цикле устанавливается флаг OCF0 (OCF2) регистра TIFR и генерируется прерывание (если оно разрешено). Кроме того, при наступлении этого события может изменяться состояние вывода OC0 (OC2) микроконтроллера. Чтобы таймер/счетчик мог управлять состоянием вывода OC0 (OC2), он должен быть сконфигурирован как выходной (соответствующий разряд регистра DDR*x* должен быть установлен в «1»).

Еще раз напоминаем, что любая операция записи в счетный регистр блокирует формирование сигнала о совпадении, если оно произойдет в следующем такте.

Регистр TCCR0 (TCCR2) предназначен для управления модулем таймера/счетчика. Формат этого регистра приведен на **Рис. 2.70**, а описание его разрядов — в **Табл. 2.79**.

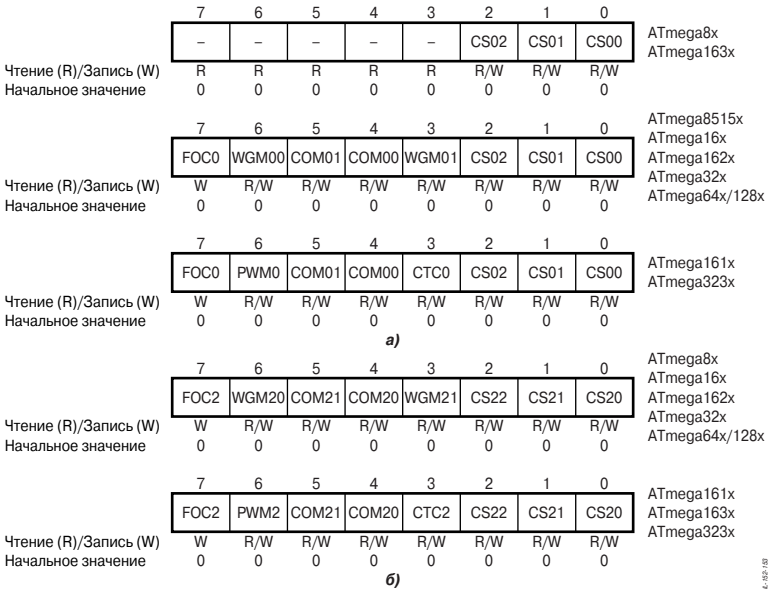


Рис. 2.70. Формат регистров TCCR0 (*а*) и TCCR2 (*б*)

Таблица 2.79. **Разряды регистра TCCR0 (TCCR2)**

<table>
<tr><th>Разряд</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 разряд состояние вывода OCn изменяется в соответствии с установкам разрядов COMn1:COMn0. Прерывание при этом не генерируется и сброс таймера (в режиме CTC) не производится. В режимах Fast PWM и Phase Correct PWM этот разряд должен быть сброшен в «0». При чтении разряда всегда возвращается «0».</td></tr>
<tr><td rowspan="6">6, 3</td><td rowspan="6">WGMn1:WGMn0 (CTCn, PWMn в ATmega161x)</td><td colspan="4">Режим работы таймера/счетчика. Эти разряды определяют режим работы таймера/счетчика следующим образом:</td></tr>
<tr><td>Номер режима</td><td>WGMn1 (CTCn)</td><td>WGMn0 (PWMn)</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>3</td><td>1</td><td>1</td><td>Fast PWM</td></tr>
<tr><td>5, 4</td><td>COMn1:COMn0</td><td colspan="4">Режим работы блока сравнения. Эти разряды определяют поведение вывода OCn при наступлении события «Совпадение». Влияние содержимого этих разрядов 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 разряды определяют источник тактового сигнала микроконтроллера. Действие этих разрядов зависит от исполнения таймера/счетчика и будет описано ниже</td></tr>
</table>

Примечания: 1. *n* = 0 или 2.

2. В регистре TCCR0 моделей ATmega8x и ATmega163x задействованы только разряды CS02...CS00. Остальные разряды регистра зарезервированы и читаются как «0». Таймер/счетчик этих моделей работает только в режиме Normal.

13.5.1. Управление тактовым сигналом

Формирование тактового сигнала таймера/счетчика clk_{T0} (clk_{T2}) осуществляется блоком предделителя, который был рассмотрен в 13.4.

В качестве тактового сигнала clk_{T0} (clk_{T2}) таймера/счетчика T0 исполнений 1, 2 и таймера/счетчика T2 исполнения 2 может использоваться (**Рис. 2.67, а**):

- системный тактовый сигнал ($clk_{T0\,(T2)} = clk_{I/O}$);
- масштабированный системный тактовый сигнал ($clk_{T0\,(T2)} = clk_{I/O}/n$);
- внешний сигнал, поступающий на вход T0 (T2) микроконтроллера ($clk_{T0\,(T2)} = clk_{EXT}$).

Тактовый сигнал таймеров/счетчиков T0 и T2 исполнения 3 может формироваться либо из системного тактового сигнала $clk_{I/O}$ ($clk_{T0\,(T2)} = clk_{I/O}/n$),

либо в асинхронном режиме из сигнала от дополнительного кварцевого резонатора ($clk_{T0\ (T2)} = clk_{TOSC1}/n$). Переключение между синхронным и асинхронным режимами работы осуществляется с помощью разряда AS0 (AS2) регистра ASSR (**Рис. 2.67, б**).

Выбор источника тактового сигнала, а также запуск и остановка таймеров/счетчиков осуществляются с помощью разрядов CS02...CS00 (CS22...CS20) регистра управления таймером TCCR0 (TCCR2) согласно **Табл. 2.80**.

Таблица 2.80. **Выбор источника тактового сигнала таймеров/счетчиков T0 и T2**

CS*n*2	CS*n*1	CS*n*0	Источник тактового сигнала		
			Исполнения 1, 2	Исполнение 3	
				AS*n* = «0»	AS*n* = «1»
0	0	0	Таймер/счетчик остановлен	Таймер/счетчик остановлен	
0	0	1	$clk_{I/O}$	$clk_{I/O}$	clk_{TOSC1}
0	1	0	$clk_{I/O}/8$	$clk_{I/O}/8$	$clk_{TOSC1}/8$
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.

13.5.2. Режимы работы

Режим работы таймера/счетчика T0 (T2) определяется состоянием разрядов WGMn1:WGMn0 регистра TCCR0 (TCCR2). Зависимость режима работы таймеров/счетчиков от состояния этих разрядов показана в **Табл. 2.81**.

Таблица 2.81. **Режимы работы таймеров/счетчиков T0 и T2**

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

Примечание: *n* = 0 или 2.

13.5.2.1. Режим Normal

Это наиболее простой режим работы таймеров/счетчиков. А в таймере/счетчике T0 моделей ATmega8x и ATmega163x это вообще единственный режим. В этом режиме счетный регистр функционирует как обычный суммирующий счетчик. По каждому импульсу тактового сигнала clkT0 (clkT2) осуществляется инкремент счетного регистра. При переходе через значение $FF возникает переполнение, и счет продолжается со значения $00. В том же такте сигнала clkT0 (clkT2), в котором обнуляется регистр TCNT0 (TCNT2), устанавливается в «1» флаг переполнения TOV0 (TOV2).

В таймерах/счетчиках исполнений 2 и 3 при равенстве счетного регистра и регистра сравнения устанавливается флаг прерывания OCF0 (OCF2) и, если разряд OCIE0 (OCIE2) регистра TIMSK установлен в «1», генерируется прерывание. Наряду с установкой флага при равенстве счетного регистра и регистра сравнения может изменяться состояние вывода OC0 (OC2) микроконтроллера. Каким образом оно будет изменяться, определяется разрядами COM01:COM00 (COM21:COM20) регистра TCCR0 (TCCR2) в соответствии с **Табл. 2.82**.

Таблица 2.82. **Управление выводом OC0 (OC2) в режиме Normal**

COM*n*1	COM*n*0	Описание
0	0	Таймер/счетчик T*n* отключен от вывода OC*n*
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 OC0 (OC2) может быть изменено принудительно, записью лог. 1 в разряд FOC0 (FOC2) регистра TCCR0 (TCCR2). Прерывание при этом не генерируется.

13.5.2.2. Режим CTC (сброс при совпадении)

В этом режиме счетный регистр тоже функционирует как обычный суммирующий счетчик, инкремент которого осуществляется по каждому импульсу тактового сигнала clk_{T0} (clk_{T2}). Однако максимально возможное значение счетного регистра и, следовательно, разрешающая способность счетчика определяется регистром сравнения OCR0 (OCR2). После достижения значения, записанного в регистре сравнения, счет продолжается со значения «$00». В том же такте сигнала clk_{T0} (clk_{T2}), в котором обнуляется счетный регистр, устанавливается флаг прерывания TOV0 (TOV2) регистра TIFR. Временные диаграммы для этого режима работы таймера/счетчика приведены на **Рис. 2.71**.

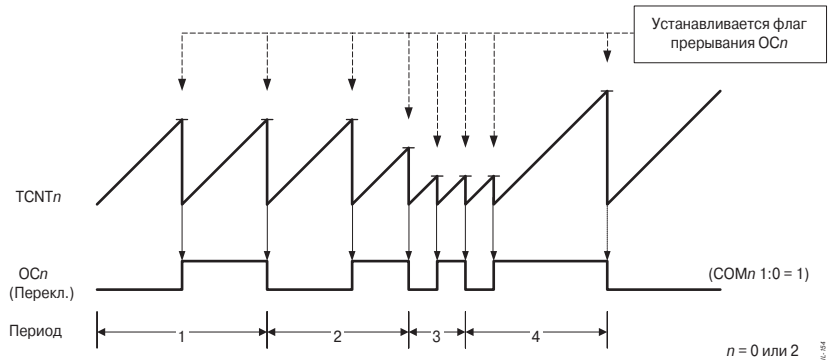


Рис. 2.71. Временные диаграммы для режима CTC

При достижении счетчиком максимального значения устанавливается флаг OCF0 (OCF2) регистра TIFR. Одновременно с установкой флага может изменяться состояние вывода OC0 (OC2) микроконтроллера. Поведение вывода определяется разрядами COM01:COM00 (COM21:COM20) регистра TCCR0 (TCCR2), как указано в **Табл. 2.83**.

Таблица 2.83. **Управление выводом OC0 (OC2) в режиме CTC**

COM*n*1	COM*n*0	Описание
0	0	Таймер/счетчик T*n* отключен от вывода OC*n*
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 разряды COM01:COM00 (COM21:COM20) значение «01» (переключение состояния вывода). Частота генерируемого сигнала будет определяться выражением $f_{OCn} = f_{clk\_I/O}/2N(1 + OCRn)$, где N — коэффициент деления предделителя (см. **Табл. 2.80**).

При необходимости состояние вывода OC0 (OC2) может быть изменено принудительно, записью лог. 1 в разряд FOC0 (FOC2) регистра TCCR0 (TCCR2). Прерывание при этом не генерируется и сброса счетного регистра не производится.

13.5.2.3. Режим Fast PWM

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

Счетный регистр в этом режиме функционирует как суммирующий счетчик, инкремент которого осуществляется по каждому импульсу тактового сигнала clk_{T0} (clk_{T2}). Состояние счетчика изменяется от \$00 до \$FF, после чего счетный регистр сбрасывается и цикл повторяется. При достижении счетчиком максимального значения устанавливается флаг прерывания TOV0 (TOV2) регистра TIFR. При равенстве содержимого счетного регистра и регистра сравнения OCR0 (OCR2) устанавливается флаг OCF0 (OCF2) регистра TIFR.

Особенностью работы схемы сравнения в этом режиме является двойная буферизация записи в регистр OCR0 (OCR2), заключающаяся в том, что записываемое число на самом деле сохраняется в специальном буферном регистре. А изменение содержимого регистра сравнения происходит только в момент достижения счетчиком максимального значения \$FF. Благодаря такому решению исключается появление несимметричных импульсов сигнала на выходе модулятора (помех), которые были бы неизбежны при непосредственной записи в регистр сравнения.

Поведение вывода OC0 (OC2) микроконтроллера в этом режиме также определяется содержимым разрядов COM01:COM00 (COM21:COM20) регистра TCCR0 (TCCR2) (**Табл. 2.84** и **Рис. 2.72**).

Таблица 2.84. **Поведение вывода OC0 (OC2) в режиме Fast PWM**

COM*n*1	COM*n*0	Поведение вывода OC*n*
0	0	Таймер/счетчик T*n* отключен от вывода OC*n*
0	1	Зарезервировано
1	0	Сбрасывается в «0» при равенстве регистров TCNT*n* и OCR*n*. Устанавливается в «1» при TCNT*n* = \$00 (неинвертированный ШИМ-сигнал)
1	1	Устанавливается в «1» при равенстве регистров TCNT*n* и OCR*n*. Сбрасывается в «0» при TCNT*n* = \$00 (инвертированный ШИМ-сигнал)

Примечание: *n* = 0 или 2.

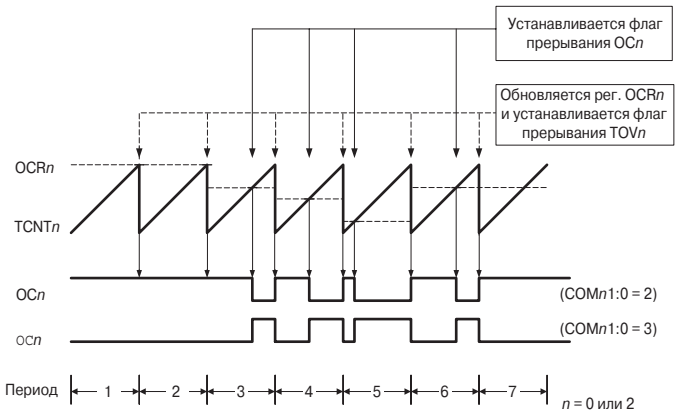


Рис. 2.72. Формирование ШИМ-сигнала в режиме Fast PWM

Частота генерируемого сигнала $f_{OCn} = f_{clk\_I/O}/256N$, где N — коэффициент деления предделителя (см. **Табл. 2.80**).

Отдельно следует рассмотреть случаи, когда в регистре сравнения находятся предельно возможные значения ($00 и $FF). В первом случае, если содержимое регистра сравнения OCR0 (OCR2) равно $00, на выходе OC0 (OC2) при каждом 256-м такте сигнала clk_{T0} (clk_{T2}) будут наблюдаться короткое выбросы. Если же содержимое регистра сравнения OCR0 (OCR2) равно $FF, то вывод OC0 (OC2) переключится в устойчивое состояние, определяемое установками разрядов COM01:COM00 (COM21:COM20).

13.5.2.4. Режим Phase Correct PWM

Режим Phase Corect 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 $FF, а затем обратно до $00. Соответственно максимальная частота сигнала в этом режиме в два раза меньше максимальной частоты сигнала в режиме Fast PWM. Тем не менее, благодаря «симметричности» изменения состояния счетчика, режим Phase Correct PWM предпочтительнее использовать для решения задач управления двигателями.

При достижении счетчиком максимального ($FF) значения происходит смена направления счета, но счетчик остается в этом состоянии в течение

одного периода сигнала clk_{T0} (clk_{T2}). При достижении счетчиком минимального значения ($00) также происходит смена направления счета и одновременно устанавливается флаг прерывания TOV0 (TOV2) регистра TIFR. При равенстве содержимого счетного регистра и регистра сравнения OCR0 (OCR2) устанавливается флаг OCF0 (OCF2) регистра TIFR и изменяется состояние вывода OC0 (OC2). Каким образом изменяется состояние вывода OC0 (OC2) определяется, как обычно, содержимым разрядов COM01:COM00 (COM21:COM20) регистра TCCR0 (TCCR2) (**Табл. 2.85** и **Рис. 2.73**).

Таблица 2.85. **Поведение вывода OC0 (OC2) в режиме Phase Correct PWM**

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

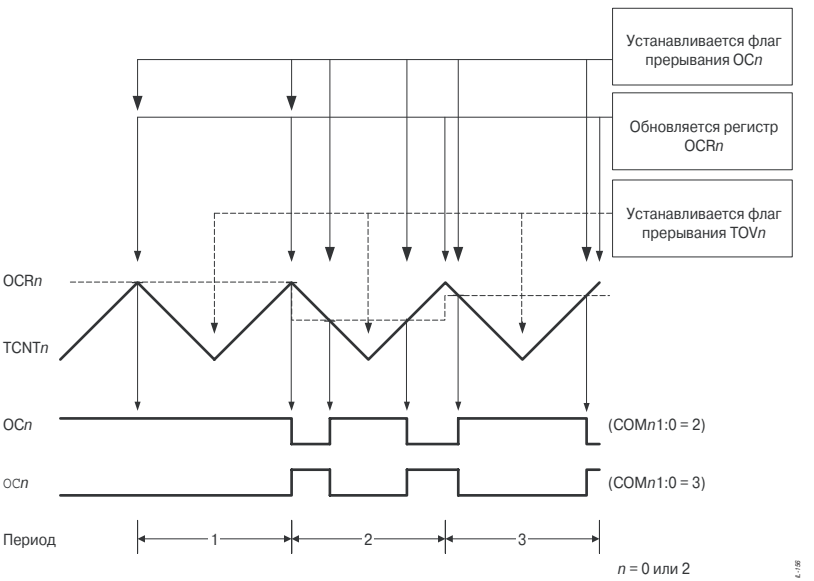


Рис. 2.73. Формирование ШИМ-сигнала в режиме Phase Correct PWM

Для избежания несимметричных выбросов в этом режиме тоже реализована двойная буферизация записи в регистр OCR0 (OCR2). Благодаря этому действительное изменение содержимого регистра сравнения происходит только в момент достижения счетчиком максимального значения $FF.

Если в регистр сравнения записать значение $00 или $FF, то при следующем совпадении состояния счетчика и содержимого регистра сравнения выход схемы сравнения переключится в устойчивое состояние согласно **Табл. 2.86**.

Таблица 2.86. **Устойчивые состояния выхода схемы сравнения**

COM*n*1	COM*n*0	Регистр OCR*n*	Состояние вывода OC*n*
1	0	$00	0
1	0	$FF	1
1	1	$00	1
1	1	$FF	0

Примечание: $n = 0$ или 2.

Частота генерируемого в рассматриваемом режиме сигнала определяется выражением $f_{OCn} = f_{clk\_I/O}/512N$, где N — коэффициент деления предделителя (см. **Табл. 2.80**).

13.5.3. 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. Задатчиком частоты сигнала может быть как кварцевый резонатор, подключаемый к выводам TOSC1 и TOSC2 микроконтроллера, так и сигнал от внешней схемы, подаваемый на вывод TOSC1. Несмотря на то что тактовый генератор таймера/счетчика настроен на частоту 32768 Гц, частота кварцевого резонатора либо сигнала от внешней схемы может лежать в пределах 0...256 кГц. При этом она должна быть в четыре раза меньше частоты тактового сигнала микроконтроллера.

Непосредственная запись в регистры TCNT0 (TCNT2), OCR0 (OCR2) и TCCR0 (TCCR2) в асинхронном режиме синхронизируется с тактовым сигналом таймера/счетчика. При записи числа в любой из указанных регистров оно сохраняется в специальном временном регистре, своем для каждого регистра таймера/счетчика. А пересылка содержимого временного регистра в

рабочий регистр таймера/счетчика осуществляется по третьему после записи положительному фронту сигнала на выводе TOSC1. Соответственно запись нового значения можно производить только после пересылки содержимого временного регистра в регистр таймера/счетчика.

Для определения момента действительного изменения регистров таймера/счетчика, а также для переключения таймера/счетчика в асинхронный режим предназначен регистр ASSR. Положение этого регистра в разных моделях микроконтроллеров указано в **Табл. 2.78**. Формат этого регистра приведен на **Рис. 2.74**, а описание отдельных его разрядов приведено в **Табл. 2.87**.

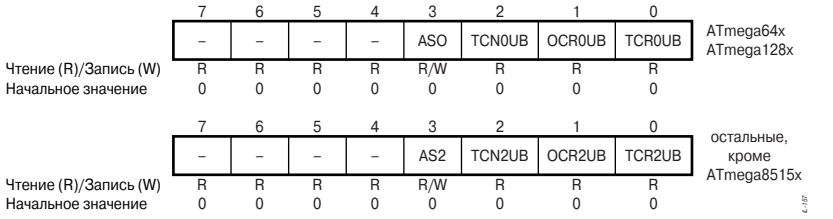


Рис. 2.74. Формат регистра ASSR

Таблица 2.87. **Разряды регистра состояния асинхронного режима ASSR**

Разряд	Название	Описание
7...4	—	Зарезервированы, читаются как «0»
3	AS*n*	**Переключение режима работы.** Если разряд установлен в «1», на вход предделителя таймера/счетчика T*n* поступают импульсы с кварцевого генератора таймера/счетчика (асинхронный режим). В этом режиме выводы TOSC1 и TOSC2 используются для подключения кварцевого резонатора и соответственно не могут использоваться как контакты ввода/вывода общего назначения. Если разряд сброшен в «0», на вход предделителя поступает внутренний тактовый сигнал микроконтроллера. В этом случае выводы TOSC1 и TOSC2 являются линиями ввода/вывода общего назначения. При изменении состояния этого разряда содержимое регистров TCNT2, OCR2 и TCCR2 может быть повреждено
2	TCN*n*UB	**Состояние обновления регистра TCNT*n***. При записи в регистр TCNT*n* этот флаг устанавливается в «1», а после пересылки записываемого значения в этот регистр, флаг аппаратно сбрасывается в «0». Таким образом, сброшенный флаг TCN*n*UB означает, 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 повреждению прежнего содержимого регистра и к генерации прерывания

Продолжение таблицы 2.87

Разряд	Название	Описание
1	OCR*n*UB	**Состояние обновления регистра OCR*n***. При записи в регистр OCR*n* этот флаг устанавливается в «1», а после пересылки записываемого значения в этот регистр флаг аппаратно сбрасывается в «0». Таким образом, сброшенный флаг OCR*n*UB означает, что регистр OCR*n* готов для записи в него нового значения. Запись в регистр OCR*n* при установленном флаге OCR*n*UB может привести к повреждению прежнего содержимого регистра и к генерации прерывания
0	TCR*n*UB	**Состояние обновления регистра TCCR*n***. При записи в регистр TCCR*n* этот флаг устанавливается в «1», а после пересылки записываемого значения в этот регистр флаг аппаратно сбрасывается в «0». Таким образом, сброшенный флаг TCR*n*UB означает, что регистр TCCR*n* готов для записи в него нового значения. Запись в регистр TCCR*n* при установленном флаге TCR*n*UB может привести к повреждению прежнего содержимого регистра и к генерации прерывания

Примечание: *n* = 0 для моделей ATmega64x/128x и *n* = 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 действовать в следующей последовательности:

- запретить прерывания от таймера/счетчика;
- переключить его в требуемый режим;
- записать новые значения в регистры TCNT2, OCR2 и TCCR2;
- в случае переключения в асинхронный режим ждать, пока флаги TCN0UB (TCN2UB), OCR0UB (OCR0UB) и TCR0UB (TCR0UB) не будут сброшены;
- сбросить флаги прерываний таймера/счетчика;
- 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 машинных цикла 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 вывода OC0 (OC2) 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 Power Down, Power Save, Standby и Extended Standby.

Первое замечание касается использования прерываний от таймера/счет-

чика для «пробуждения» микроконтроллера. Если перевод микроконтроллера в режим Power Save или Extended Standby осуществляется сразу 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 регистр TCNT0 (TCNT2) или OCR (OCR2) работа блока сравнения заблокирована. Соответственно, если переход в спящий режим произойдет до окончания операции записи в указанные регистры, прерывания от схемы сравнения никогда не произойдет 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 повторным переходом в спящий режим будет меньше указанного, генерации прерывания и соответственно перехода микроконтроллера в рабочий режим не произойдет. 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 событий. 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 потерянным (из-за нестабильности тактового сигнала во время запуска генератора). Причем это справедливо не только при использовании кварцевого резонатора, но и при использовании внешнего тактового сигнала.

13.6. Таймеры/счетчики T1 и T3

Шестнадцатиразрядный таймер/счетчик T1 присутствует во всех моделях микроконтроллеров семейства Mega, а таймер/счетчик T3 — только в моделях ATmega162x (отсутствует в режиме совместимости с ATmega161x) и ATmega64x/ATmega128x. Как и таймеры/счетчики T0 и T2, они могут использоваться для формирования временных интервалов, для подсчета числа внешних событий, формирования сигналов и генерации сигналов с ШИМ (но уже переменной разрядности). В дополнение к этому таймеры/счетчики

T1/T3 могут по внешнему сигналу сохранять свое текущее состояние в отдельном регистре ввода/вывода.

Таймеры/счетчики T1/T3 различных моделей отличаются только количеством блоков сравнения и соответственно количеством каналов генерации ШИМ-сигналов. Так, если в моделях ATmega64x/ATmega128x таймеры/счетчики T1/T3 имеют по три блока сравнения, то в остальных моделях микроконтроллеров — только по два. Упрощенная структурная схема самых развитых таймеров (модели ATmega64x/ATmega128x) приведена на **Рис. 2.75**.

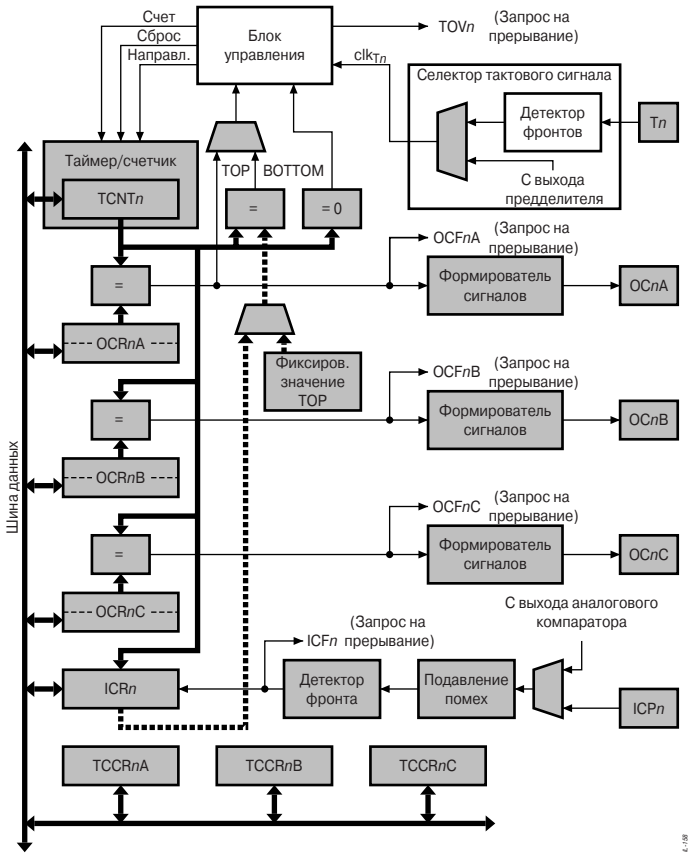


Рис. 2.75. Структурная схема 16-разрядных таймеров/счетчиков (T1 и T3)

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

- 16-разрядный счетный регистр TCNT1 (TCNT3);
- 16-разрядный регистр захвата ICR1 (ICR3);
- два или три 16-разрядных регистра сравнения OCR1A, OCR1B, OCR1C (OCR3A, OCR3B, OCR3C);
- два или три 8-разрядных регистра управления TCCR1A, TCCR1B, TCCR1C (TCCR3A, TCCR3B, TCCR3C).

Каждый из 16-разрядных регистров физически размещается в двух регистрах ввода/вывода, названия которых получаются добавлением к названию регистра буквы «H» (старший байт) и «L» (младший байт). Счетный регистр таймера счетчика TCNT1, например, размещается в регистрах TCNT1H:TCNT1L.

Адреса всех регистров таймеров/счетчиков T1 и T3 указаны в **Табл. 2.88**.

Таблица 2.88. **Регистры 16-разрядных таймеров/счетчиков**

Регистр	Адрес	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega32x	ATmega323x	ATmega64x	ATmega128x
TCCR1A	$2F ($4F)	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
TCCR1B	$2E ($4E)	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
TCCR1C	($7A)	—	—	—	—	—	—	—	—	♦	♦
TCNT1	$2D:$2C ($4D:$4C)	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
OCR1A	$2B:$2A ($4B:$4A)	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
OCR1B	$29:$28 ($49:$48)	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
OCR1C	($79:$78)	—	—	—	—	—	—	—	—	♦	♦
ICR1	$27:$26 ($47:$46)	♦	—	♦	—	—	♦	♦	♦	♦	♦
	$25:$24 ($45:$44)	—	♦	—	♦	♦	—	—	—	—	—
TCCR3A	($8B)	—	—	—	—	♦	—	—	—	♦	♦
TCCR3B	($8A)	—	—	—	—	♦	—	—	—	♦	♦
TCCR3C	($8C)	—	—	—	—	♦	—	—	—	♦	♦
TCNT3	($89:$88)	—	—	—	—	♦	—	—	—	♦	♦
OCR3A	($87:$86)	—	—	—	—	♦	—	—	—	♦	♦
OCR3B	($85:$84)	—	—	—	—	♦	—	—	—	♦	♦
OCR3C	($83:$82)	—	—	—	—	—	—	—	—	♦	♦
ICR3	($81:$80)	—	—	—	—	♦	—	—	—	♦	♦

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

- при переполнении счетного регистра;
- при равенстве счетного регистра и регистра сравнения (по одному прерыванию на каждый блок сравнения);
- при сохранении счетного регистра в регистре захвата.

Флаги всех прерываний таймеров/счетчиков T1 и T3 находятся в регистрах TIFR и ETIFR, а разрешение/запрещение этих прерываний осуществляется установкой/сбросом соответствующих флагов регистров TIMSK и ETIMSK (см. подраздел 13.3).

Счетный регистр таймера/счетчика TCNT1 (3) входит в состав основного блока модуля — блока реверсивного счетчика. В зависимости от режима работы модуля содержимое счетного регистра сбрасывается, инкрементируется или декрементируется по каждому импульсу тактового сигнала таймера/счетчика clk_{T1} (clk_{T3}). Независимо от того, присутствует тактовый сигнал или нет, регистр доступен в любой момент времени как для чтения, так и для записи. При этом любая операция записи в счетный регистр блокирует работу всех блоков сравнения на время одного периода тактового сигнала таймера/счетчика. После подачи напряжения питания в регистре TCNT1 (3) находится нулевое значение. При некоторых изменениях состояния таймера/счетчика, определяемых режимом его работы, устанавливается флаг TOV1 (3) регистра TIFR. Разрешение прерывания осуществляется установкой в «1» разряда TOIE1 (3) регистра TIMSK.

Регистры OCR1A/OCR1B/OCR1C (OCR3A/OCR3B/OCR3C) входят в состав блоков сравнения. Во время работы таймера/счетчика производится непрерывное (в каждом машинном цикле) сравнение этих регистров с регистром TCNT1 (TCNT3). В случае равенства содержимого регистра сравнения и счетного регистра в следующем машинном цикле устанавливается соответствующий флаг OCF1A/OCF1B/OCF1C (OCF3A/OCF3B/OCF3C) регистра TIFR и генерируется прерывание (если оно разрешено). Также при наступлении этого события может изменяться состояние вывода OC1A/OC1B/OC1C (OC3A/OC3B/OC3C) микроконтроллера. Чтобы таймер/счетчик мог управлять состоянием какого-либо из этих выводов, он должен быть сконфигурирован как выходной (соответствующий разряд регистра DDR*x* должен быть установлен в «1»).

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

Регистр захвата ICR1 (ICR3) входит в состав блока захвата, назначе-

ние которого — сохранение в определенный момент времени состояния таймера/счетчика в регистре захвата ICR1 (ICR3). Это действие может производиться либо по активному фронту сигнала на выводе ICP1 (ICP3) микроконтроллера, либо (для таймера/счетчика T1) по сигналу от аналогового компаратора. Одновременно с записью в регистр захвата устанавливается флаг ICF1 регистра TIFR (ICF3 регистра ETIFR) и генерируется запрос на прерывание. Разрешение прерывания осуществляется установкой в «1» разряда TICIE1 регистра TIMSK (TICIE3 регистра ETIMSK). Программно запись в регистр ICR1 (ICR3) возможна только в режимах, в которых регистр захвата определяет модуль счета таймера/счетчика (**Табл. 2.93**). Вывод ICP1 (ICP3) в этих режимах отключен и функция захвата, соответственно, выключена.

Упрощенная структурная схема блока захвата приведена на **Рис. 2.76**.

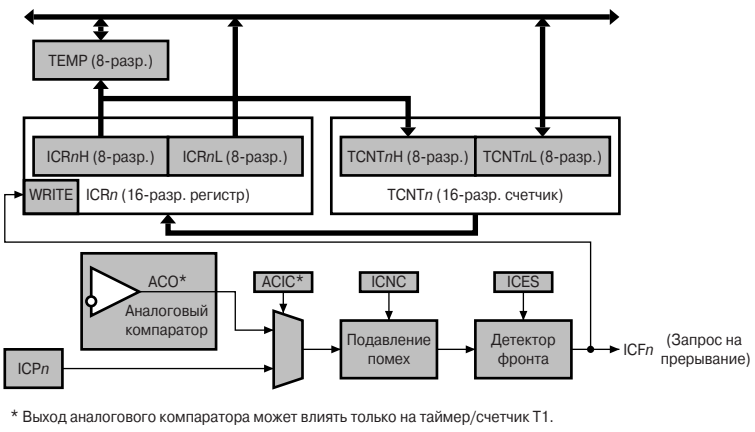


Рис. 2.76. Структурная схема блока захвата

Для управления схемой захвата используются два разряда регистра TCCR1B (TCCR3B) — ICNC1 (ICNC3) и ICES1 (ICES3). Разряд ICNC1 (ICNC3) управляет схемой подавления помех. Если этот разряд сброшен в «0», схема подавления помех выключена и захват производится по первому же активному фронту на выходе мультиплексора (**Рис. 2.76**). Если же этот разряд установлен в «1», то при появлении активного фронта производится 4 выборки с частотой, равной тактовой частоте микроконтроллера. Захват будет выполнен только в том случае, если все выборки имеют уровень, соответствующий активному фронту сигнала (лог. 1 для нарастающего и лог. 0 для спадающего).

Активный фронт сигнала, т. е. фронт, по которому будет выполнено сохранение содержимого счетного регистра в регистре захвата, определяется состоянием разряда ICES1 (ICES3). Если этот разряд сброшен в «0», то активным является спадающий фронт. Если разряд установлен в «1», то активным является нарастающий фронт. Для захвата по сигналу с вывода ICP1 (ICP3), этот вывод должен быть сконфигурирован как входной (разряд регистра DDR*x*, соответствующий выводу, должен быть сброшен в «0»). Если же он будет сконфигурирован как выходной, захват можно будет осуществлять программно, управляя соответствующим разрядом порта.

Следует понимать, что между изменением состояния входа блока захвата и копированием счетного регистра в регистр захвата таймера/счетчика проходит некоторое время. Эту задержку вносит каскад, состоящий из синхронизатора (на рисунке не показан) и детектора фронта. Величина задержки составляет 2.5...3.5 машинных циклов. При включении схемы подавления помех задержка увеличивается еще на 4 машинных цикла.

Для управления таймером/счетчиком используются три регистра управления: TCCR1A (TCCR3A), TCCR1B (TCCR3B), TCCR1C (TCCR3C). Формат этих регистров приведен на **Рис. 2.77...2.79**, а описание их разрядов — в **Табл. 2.89...2.91**.

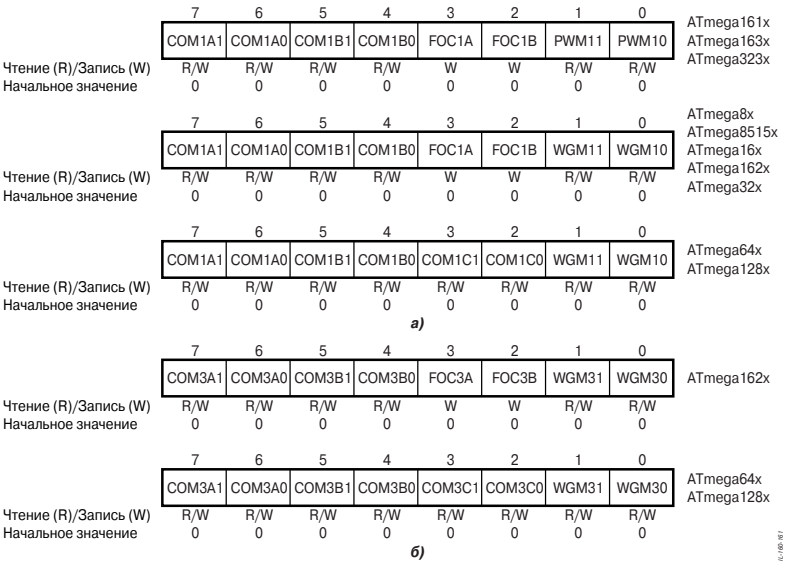


Рис. 2.77. Формат регистров TCCR1A (*а*) и TCCR3A (*б*)

Таблица 2.89. **Разряды регистра TCCR1A (TCCR3A)**

Разряд	Название	Описание
7, 6	COM*n*A1:COM*n*A0	**Режим работы блока сравнения *x*.** Эти разряды определяют поведение вывода OC*nx* при наступлении события «Совпадение». Влияние содержимого этих разрядов на состояние вывода зависит от режима работы таймера/счетчика
5, 4	COM*n*B1:COM*n*B0	
3, 2	COM*n*C1:COM*n*C0	
1, 2	WGM*n*1:WGM*n*0	**Режим работы таймера/счетчика.** Совместно с разрядами WGM*n*3:WGM*n*2 регистра TCCR*n*B определяют режим работы таймера/счетчика T*n* (см. **Табл. 2.83**)

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

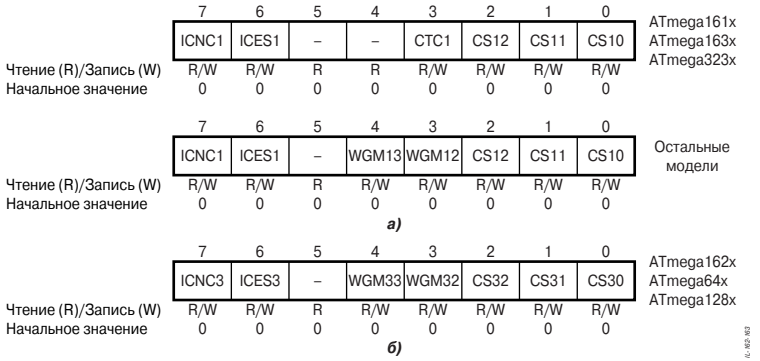


Рис.2.78. Формат регистров TCCR1B (*а*) и TCCR3B (*б*)

Таблица 2.90. **Разряды регистра TCCR1B (TCCR3B)**

Разряд	Название	Описание
7	ICNC*n*	**Управление схемой подавления помех блока захвата.** Если разряд сброшен в «0», схема подавления помех выключена (захват производится по первому активному фронту). Если разряд установлен в «1», схема подавления помех включена и захват осуществляется только в случае 4-х одинаковых выборок, соответствующих активному фронту сигнала

Продолжение таблицы 2.90

Разряд	Название	Описание
6	ICES*n*	**Выбор активного фронта сигнала захвата.** Если разряд ICES*n* сброшен в «0», сохранение счетного регистра в регистре захвата осуществляется по спадающему фронту сигнала. Если разряд установлен в «1», сохранение счетного регистра в регистре захвата осуществляется по нарастающему фронту сигнала. Одновременно с сохранением счетного регистра устанавливается также флаг прерывания ICF*n* регистра TIFR (ETIFR)
5	—	Не используется, читается как «0»
4, 3	WGM*n*3:WGM*n*2	**Режим работы таймера/счетчика.** Совместно с разрядами WGM*n*1:WGM*n*0 регистра TCCR*n*Ф определяют режим работы таймера/счетчика T*n* (**Табл. 2.83**)
2...0	CS*n*2...CS*n*0	**Управление тактовым сигналом.** Эти разряды определяют источник тактового сигнала микроконтроллера

Примечание: *n* = 1 или 3.

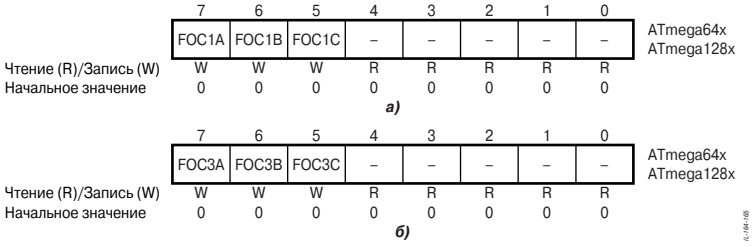


Рис. 2.79. Формат регистров TCCR1C (*а*) и TCCR3C (*б*)

Таблица 2.91. **Разряды регистра TCCR1C (TCCR3C)**

Разряд	Название	Описание
7	FOC*n*A	**Принудительное изменение состояния вывода OC*nx***. При записи в разряд FOC*nx* лог. 1, состояние вывода OC*nx* изменяется в соответствии с установками разрядов COM*n*1*x*:COM*n*0*x* регистра TCCR*n*A. Прерывание при этом не генерируется и сброс таймера (в режиме CTC) не производится. Эта функция доступна только в тех режимах, которые не используются для генерации сигнала с ШИМ. При чтении разряда всегда возвращается «0»
6	FOC*n*B	
5	FOC*n*C	
4...0	—	Не используется, читается как «0»

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

13.6.1. Обращение к 16-разрядным регистрам

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

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

Для выполнения цикла чтения 16-разрядного регистра первым должен быть прочитан младший байт. При его чтении содержимое старшего байта помещается в регистр TEMP. При последующем чтении старшего байта возвращается значение, сохраненное в регистре TEMP. Исключение составляют только регистры сравнения OCR1A/B/C (OCR3A/B/C), при чтении которых регистр TEMP не задействуется.

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

13.6.2. Управление тактовым сигналом

Формирование тактового сигнала таймера/счетчика clk_{T1}/clk_{T3} осуществляется блоком предделителя, который был рассмотрен в параграфе 6.4 этой главы.

В качестве тактового сигнала $clk_{T1}(clk_{T3})$ таймеров/счетчиков T1 и T3 может использоваться (см. **Рис. 2.67, а**):

- системный тактовый сигнал ($clk_{T1(T3)} = clk_{I/O}$);

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

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 разрядов CS02...CS00 (CS22...CS20) регистра управления таймером TCCR1B (TCCR3B) согласно **Табл. 2.92**.

Таблица 2.92. **Выбор источника тактового сигнала таймеров/счетчиков T1 и T3**

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: $n = 1$ или 3.

13.6.3. Режимы работы

Режим работы таймера/счетчика T1 (T3) определяется состоянием разрядов WGM*n*3:WGM*n*2 (CTC*n* в моделях ATmega161x/163x/323x) регистра TCCR1B (TCCR3B) и разрядов WGM*n*1:WGM*n*0 (PWM*n*1:0 в моделях ATmega161x/163x/323x) регистра TCCR1A (TCCR3A). Зависимость режима работы таймеров/счетчиков от состояния этих разрядов показана в **Табл. 2.93**. В моделях ATmega161x/163x/323x режимы 8...15 отсутствуют.

Таблица 2.93. **Режимы работы таймеров/счетчиков T1 и T3**

Номер режима	WGM*n*3	WGM*n*2 (CTC1)*	WGM*n*1 (PWM11)*	WGM*n*0 (PWM10)*	Режим работы таймера/счетчика T*n*	Модуль счета (TOP)
0	0	0	0	0	Normal	$FFFF
1	0	0	0	1	Phase correct PWM, 8-разрядный	$00FF
2	0	0	1	0	Phase correct PWM, 9-разрядный	$01FF
3	0	0	1	1	Phase correct PWM, 10-разрядный	$03FF
4	0	1	0	0	CTC (сброс при совпадении)	OCR*n*A
5	0	1	0	1	Fast PWM, 8-разрядный	$00FF
6	0	1	1	0	Fast PWM, 9-разрядный	$01FF
7	0	1	1	1	Fast PWM, 10-разрядный	$03FF
8**	1	0	0	0	Phase and Frequency Correct PWM	ICR*n*
9**	1	0	0	1	Phase and Frequency Correct PWM	OCR*n*A
10**	1	0	1	0	Phase correct PWM	ICR*n*
11**	1	0	1	1	Phase correct PWM	OCR*n*A
12**	1	1	0	0	CTC (сброс при совпадении)	ICR*n*
13**	1	1	0	1	Зарезервировано	—
14**	1	1	1	0	Fast PWM	ICR*n*
15**	1	1	1	1	Fast PWM	OCR*n*A

* В моделях ATmega161x/163x/323x.
** В моделях ATmega161x/163x/323x эти режимы отсутствуют.

Примечание: *n* = 1 или 3.

13.6.3.1. 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_{T1} (clk_{T3}) осуществляется инкремент счетного регистра. При переходе через значение $FFFF возникает переполнение, и счет продолжается со значения $0000. В том же такте сигнала clk_{T1} (clk_{T3}), в котором обнуляется регистр TCNT1 (TCNT3), устанавливается в «1» флаг прерывания по переполнению TOV1 (TOV3).

Блоки сравнения обоих таймеров в этом режиме могут использоваться как для генерации прерываний, так и для формирования сигналов. Поведение выходов OC1A/OC1B/OC1C (OC3A/OC3B/OC3C) каждого из блоков сравнения таймеров/счетчиков T1 (T3) определяется состоянием разрядов COM*nX*1:COM*nX*0 регистров TCCR1A (TCCR3A), как показано в **Табл. 2.94**.

Таблица 2.94. **Управление выводом OC1A/OC1B/OC1C (OC3A/OC3B/OC3C) в режиме Normal**

COM*n*A1/COM*n*B1/ COM*n*C1	COM*n*A0/COM*n*B0/ COM*n*C0	Описание
0	0	Таймер/счетчик T*n* отключен от вывода OC*n*A/OC*n*B/OC*n*C
0	1	Состояние вывода меняется на противоположное
1	0	Вывод сбрасывается в «0»
1	1	Вывод устанавливается в «1»

Примечание: *n* = 1 или 3.

Состояние выхода любого блока сравнения также может быть изменено принудительно, записью лог. 1 в разряд FOC1A/FOC1B/FOC1C (FOC3A/FOC3B/FOC3C) регистра TCCR1C (TCCR3C). Прерывание при этом не генерируется.

13.6.3.2. Режим CTC (сброс при совпадении)

В этом режиме счетный регистр тоже функционирует как обычный суммирующий счетчик, инкремент которого осуществляется по каждому импульсу тактового сигнала clk_{T1} (clk_{T3}). Однако максимально возможное значение счетного регистра и, следовательно, разрешающая способность счетчика определяется либо регистром сравнения блока «A» OCR1A (OCR3A), либо регистром захвата ICR1 (ICR3). После достижения максимального значения счет продолжается со значения $0000. В том же такте сигнала clk_{T1} (clk_{T3}), в котором обнуляется счетный регистр, устанавливается

флаг прерывания TOV1 (TOV3) регистра TIFR (ETIFR). Временные диаграммы для этого режима работы таймера/счетчика приведены на **Рис. 2.80**.

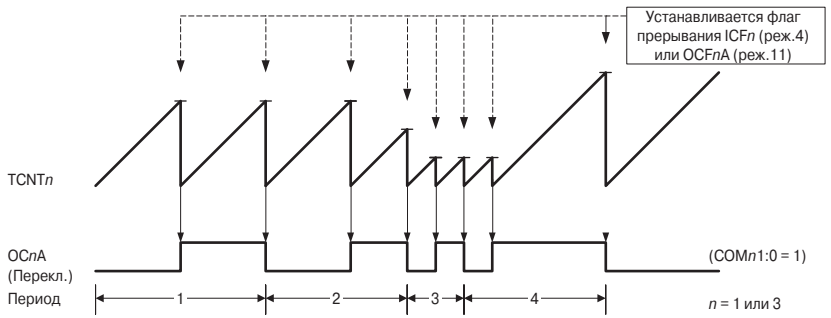


Рис. 2.80. Временные диаграммы для режима СТС

При достижении счетчиком максимального значения устанавливается флаг:

- OCF1A (OCF3A) регистра TIFR (ETIFR), если модуль счета определяется регистром сравнения OCR1A (OCR3A);
- ICF1 (ICF3) регистра TIFR (ETIFR), если модуль счета определяется регистром захвата ICR1 (ICR3).

Одновременно с установкой флага может изменяться состояние вывода OC1A (OC3A) микроконтроллера. Поведение вывода определяется состоянием разрядов COM*n*A1:COM*n*A0 регистров TCCR1A (TCCR3A), как указано в **Табл. 2.95**.

Таблица 2.95. **Управление выводом OC1A (OC3A) в режиме СТС**

COM*n*A1	COM*n*A0	Описание
0	0	Таймер/счетчик T*n* отключен от вывода OC*n*
0	1	Состояние вывода меняется на противоположное
1	0	Вывод сбрасывается в «0»
1	1	Вывод устанавливается в «1»

Примечание: *n* = 1 или 3.

Для генерации сигнала заданной частоты необходимо записать в разряды COM1A1:COM1A0 (COM3A1:COM3A0) значение «01» (переключение состояния вывода). Частота генерируемого сигнала будет определяться выражением $f_{OCn} = f_{clk\_I/O}/2N(1 + X)$, где X — модуль счета (значение, находящееся в регистре OCR1A (OCR3A) или ICR1 (ICR3); N — коэффициент деления предделителя (см. **Табл. 2.92**).

Как и в режиме Normal, состояние вывода OC1A (OC3A) при необходимости может быть изменено принудительно, записью лог. 1 в разряд FOC1A (FOC3A) регистра TCCR1C (TCCR3C). Прерывание при этом не генерируется и сброс счетного регистра не производится.

13.6.3.3. Режим Fast PWM

Режим Fast PWM («Быстродействующий ШИМ») позволяет генерировать высокочастотный сигнал с широтно-импульсной модуляцией. Этот режим практически полностью идентичен одноименному режиму 8-разрядных таймеров/счетчиков. Отличие заключается только в том, что таймеры/счетчики T1 и T3 позволяют генерировать ШИМ-сигнал различной разрядности.

Счетный регистр в этом режиме функционирует как суммирующий счетчик, инкремент которого осуществляется по каждому импульсу тактового сигнала clk_{T1} (clk_{T3}). Состояние счетчика изменяется от $0000 до максимального значения, после чего счетный регистр сбрасывается и цикл повторяется. В зависимости от установок разрядов WGM*n*3:0 (CTC*n* и PWM*n*1:0 в моделях ATmega161x/163x/323x) максимальное значение счетчика (разрешение ШИМ-сигнала) является либо фиксированным значением, либо определяется содержимым определенных регистров таймера/счетчика (**Табл. 2.96**). Разрешающая способность определяется выражением $R = \log(TOP + 1)/\log 2$, где *TOP* — модуль счета.

Таблица 2.96. **Разрешающая способность модулятора в режиме Fast PWM**

Номер режима	WGM*n*3	WGM*n*2 (CTC1)*	WGM*n*1 (PWM11)*	WGM*n*0 (PWM10)*	Разрешающая способность	Модуль счета (TOP)
5	0	1	0	1	8 разрядов	$00FF
6	0	1	1	0	9 разрядов	$01FF
7	0	1	1	1	10 разрядов	$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)

* В моделях ATmega161x/163x/323x.
** В моделях ATmega161x/163x/323x отсутствуют.

Примечание: *n* = 1 для таймера/счетчика T1, *n* = 3 для таймера/счетчика T3.

При работе с какими-либо фиксированными значениями модуля счета для задания его рекомендуется использовать регистр захвата. При этом

регистр OCR1A (OCR3A) 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. 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 TOV1 (TOV3) регистра TIFR (ETIFR). Одновременно с ним устанавливается флаг ICF*n* (режим 14) либо OCF*n*A (режим 15).

При равенстве содержимого счетного регистра и какого-либо регистра сравнения устанавливается соответствующий флаг OCF1A/OCF1B/OCF1C (OCF3A/OCF3B/OCF3C) регистра TIFR. Одновременно изменяется состояние выхода блока сравнения OC*n*A/OC*n*B/OC*n*C. Поведение этих выходов определяется содержимым разрядов COM*nX*1:COM*nX*0 регистров TCCR1A (TCCR3A) (**Табл. 2.97**). Временные диаграммы для случая, когда модуль счета определяется содержимым регистра ICR*n*A или OCR*n*A, показаны на **Рис. 2.74.**

Таблица 2.97. **Поведение вывода OC1A/OC1B/OC1C (OC3A/OC3B/OC3C) в режиме Fast PWM**

COM*n*A1/COM*n*B1/ COM*n*C1	COM*n*A0/COM*n*B0/ COM*n*C0	Описание
0	0	Таймер/счетчик T*n* отключен от вывода OC*n*A/OC*n*B/OC*n*C
0	1	WGM*n*3 = «0»: таймер/счетчик T*n* отключен от вывода OC*n*A/OC*n*B/OC*n*C*; WGM*n*3 = «1»: состояние вывода OC*n*A меняется на противоположное
1	0	Сбрасывается в «0» при равенстве счетного регистра и соответствующего регистра сравнения. Устанавливается в «1» при достижении счетчиком максимального значения (неинвертированный ШИМ-сигнал)
1	1	Устанавливается в «1» при равенстве счетного регистра и соответствующего регистра сравнения. Сбрасывается в «0» при достижении счетчиком максимального значения (неинвертированный ШИМ-сигнал)
* Также для моделей ATmega161x/163x/323x.		

Примечание: *n* = 1 или 3.

Если содержимое регистра сравнения равно модулю счета, то выход соответствующего блока сравнения переключится в устойчивое состояние, определяемое установками разрядов COM*nX*1:COM*nX*0 (**Рис. 2.81**).

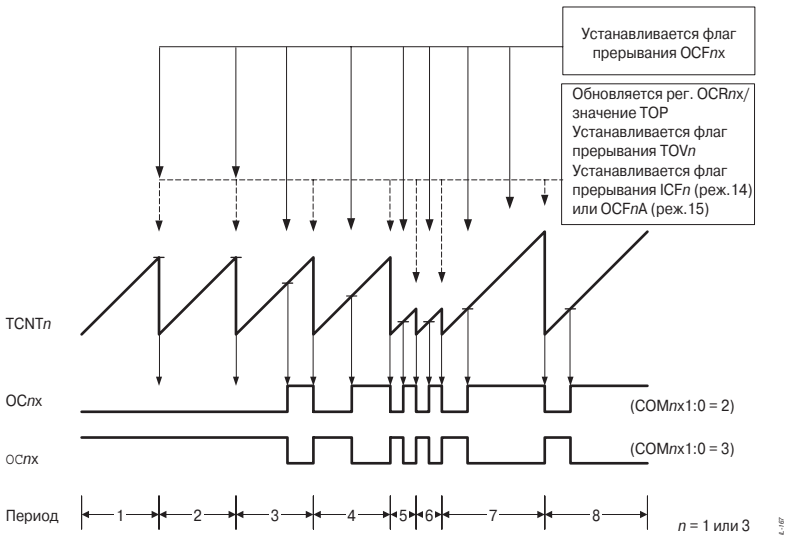


Рис. 2.81. Формирование ШИМ-сигнала в режиме Fast PWM

Частота генерируемого в этом режиме сигнала $f_{OCn} = f_{clk\_I/O}/(N \cdot TOP)$, где *TOP* — модуль счета, а *N* — коэффициент деления предделителя (см. **Табл. 2.80**).

При необходимости блок сравнения «A» в этом режиме может также использоваться для генерации сигнала меандра. Для этого необходимо записать в разряды COM1A1:COM1A0 (COM3A1:COM3A0) значение «01», задающее переключение состояния вывода OC1A (OC3A) при наступлении события «Совпадение».

13.6.3.4. Режим Phase Correct PWM

Режим Phase Corect 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 два раза меньше максимальной частоты сигнала в режиме Fast PWM.

В зависимости от установок разрядов WGM*n*3:0 (CTC*n* и PWM*n*1:0 в моделях ATmega161x/163x/323x) максимальное значение счетчика (разрешение ШИМ-сигнала) является либо фиксированным значением, либо определяется содержимым определенных регистров таймера/счетчика (**Табл. 2.98**). Разрешающая способность определяется выражением R = log(*TOP* + 1)/log2, где *TOP* — модуль счета.

Таблица 2.98. **Разрешающая способность модулятора в режиме Phase Correct PWM**

Номер режима	WGM*n*3	WGM*n*2 (CTC1)*	WGM*n*1 (PWM11)*	WGM*n*0 (PWM10)*	Разрешающая способность	Модуль счета (*TOP*)
1	0	0	0	1	8 разрядов	$00FF
2	0	0	1	0	9 разрядов	$01FF
3	0	0	1	1	10 разрядов	$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)

* В моделях ATmega161x/163x/323x.
** В моделях ATmega161x/163x/323x эти режимы отсутствуют.

Примечания: *n* = 1 для таймера/счетчика T1 и *n* = 3 для таймера/счетчика T3.

Как и в режиме Fast PWM, при работе с какими-либо фиксированными значениями модуля счета для его задания рекомендуется использовать регистр захвата. При этом регистр OCR1A (OCR3A) 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_{T1} (clk_{T3}). В этом же такте производится обновление содержимого регистра сравнения. Если модуль счета определяется регистром 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 TOV1 (TOV3) регистра TIFR (ETIFR). При равенстве содержимого счетного регистра и какого-либо регистра сравнения устанавливается

соответствующий флаг OCF1A/OCF1B/OCF1C (OCF3A/OCF3B/OCF3C) регистра TIFR. Одновременно изменяется состояние выхода блока сравнения OC*n*A/OC*n*B/OC*n*C. Как обычно, поведение вывода определяется содержимым разрядов COM*nX*1:COM*nX*0 регистров TCCR1A (TCCR3A) (**Табл. 2.99**). Временные диаграммы для случая, когда модуль счета определяется содержимым регистра ICR*n*A или OCR*n*A, показаны на **Рис. 2.82**.

Таблица 2.99. **Поведение вывода OC1A/OC1B/OC1C (OC3A/OC3B/OC3C) в режиме Phase Correct PWM**

COM*n*A1/COM*n*B1/ COM*n*C1	COM*n*A0/COM*n*B0/ COM*n*C0	Описание
0	0	Таймер/счетчик T*n* отключен от вывода OC*n*A/OC*n*B/OC*n*C
0	1	WGM*n*3 = «0»: таймер/счетчик T*n* отключен от вывода OC*n*A/OC*n*B/OC*n*C*; WGM*n*3 = «1»: состояние вывода OC*n*A меняется на противоположное
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)
* Также для моделей ATmega161x/163x/323x.		

Примечание: *n* = 1 или 3.

Следует понимать, что при изменении модуля счета во время работы таймера/счетчика, на выходе блоков сравнения могут появиться несимметричные (относительно середины периода модуляции) импульсы. Поскольку обновление содержимого регистра сравнения происходит в момент достижения счетчиком максимального значения, период ШИМ-сигнала равен 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 обратного счета также отличаются. Результатом этого и являются несимметричные импульсы на выходе блоков сравнения, как показано на **Рис. 2.82** (3-й период сигнала).

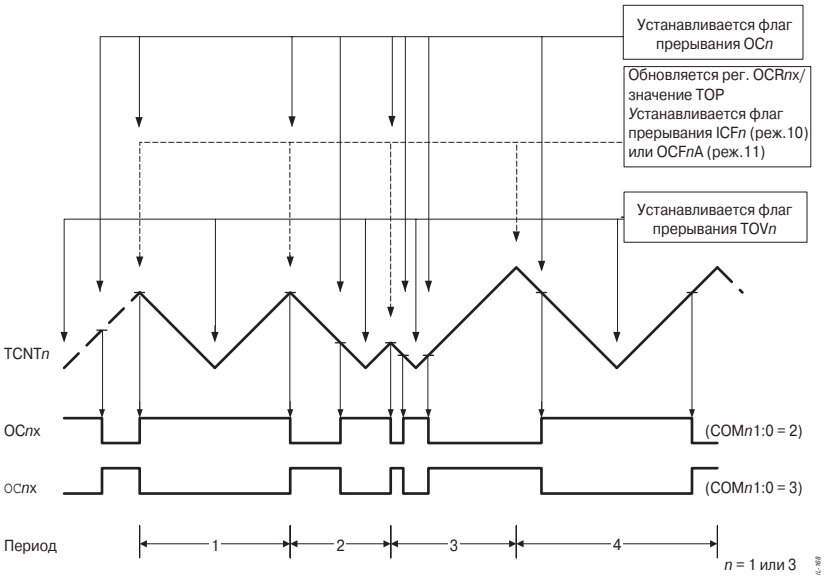


Рис. 2.82. Формирование ШИМ-сигнала в режиме Phase Correct PWM

Поэтому при частом изменении модуля счета во время работы таймера/счетчика рекомендуется использовать режим Phase and Frequency Correct PWM, описанный в следующем параграфе. Если же используется постоянное значение модуля счета, 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. 2.100**.

Таблица 2.100. **Устойчивые состояния выхода схемы сравнения**

COM*n*A1/COM*n*B1/ COM*n*C1	**COM*n*A0/COM*n*B0/ COM*n*C0**	**Регистр OCR*n*A/ OCR*n*B/OCR*n*C**	**Состояние вывода OC*n*A/OC*n*B/OC*n*C**
1	0	$0000	0
1	0	TOP	1
1	1	$0000	1
1	1	TOP	0

Примечание: *n* = 1 или 3.

Частота генерируемого сигнала $f_{OCn} = f_{clk\_I/O}/(2N \cdot TOP)$, где *TOP* — модуль счета, а *N* — коэффициент деления предделителя, приведенный в **Табл. 2.80**.

13.6.3.5. Режим Phase and Frequency Correct PWM

Режим Phase and Frequency Corect PWM («ШИМ с точной фазой и частотой»), отсутствующий в моделях ATmega161x/163x/323x, очень похож на режим Phase Corect PWM. Единственная принципиальная разница между ними — момент обновления содержимого регистра сравнения.

Максимальное значение счетчика (разрешение ШИМ-сигнала) в этом режиме может определяться только регистрами ICR1A (ICR3A) или OCR1A (OCR3A) таймера/счетчика T1 (T3), как показано в **Табл. 2.101**. Разрешающая способность определяется выражением $R = \log(TOP + 1)/\log 2$, где *TOP* — модуль счета.

Таблица 2.101. **Разрешающая способность модулятора в режиме Phase and Frequency Correct PWM**

Номер режима	WGM*n*3	WGM*n*2	WGM*n*1	WGM*n*0	Разрешающая способность	Модуль счета (TOP)
8	1	0	1	0	Переменная (2...16)	ICR*n*A ($0003...$FFFF)
9	1	0	1	1	Переменная (2...16)	OCR*n*A ($0003...$FFFF)

Примечания: *n* = 1 для таймера/счетчика T1, *n* = 3 для таймера/счетчика T3.

Как и в остальных режимах, при работе с какими-либо фиксированными значениями модуля счета, для задания его рекомендуется использовать регистр захвата. При этом регистр OCR1A (OCR3A) 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_{T1} (clk_{T3}). В этом же такте устанавливается флаг ICF*n* либо OCF*n*A (зависит 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 TOV1 (TOV3) регистра TIFR (ETIFR) и производится обновление содержимого регистра сравнения.

При равенстве содержимого счетного регистра и какого-либо регистра сравнения устанавливается соответствующий флаг OCF1A/OCF1B/OCF1C

(OCF3A/OCF3B/OCF3C) регистра TIFR. Одновременно изменяется состояние выхода блока сравнения OC*n*A/OC*n*B/OC*n*C. Как обычно, поведение вывода определяется содержимым разрядов COM*nX*1:COM*nX*0 регистров TCCR1A (TCCR3A) (**Табл. 2.102** и **Рис. 2.83**).

Таблица 2.102. **Поведение вывода OC1A/OC1B/OC1C (OC3A/OC3B/OC3C) в режиме Phase and Frequency Correct PWM**

COM*n*A1/COM*n*B1/ COM*n*C1	COM*n*A0/COM*n*B0/ COM*n*C0	Описание
0	0	Таймер/счетчик T*n* отключен от вывода OC*n*A/OC*n*B/OC*n*C
0	1	WGM*n*3 = «0»: таймер/счетчик T*n* отключен от вывода OC*n*A/OC*n*B/OC*n*C; WGM*n*3 = «1»: состояние вывода OC*n*A меняется на противоположное
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 или 3.

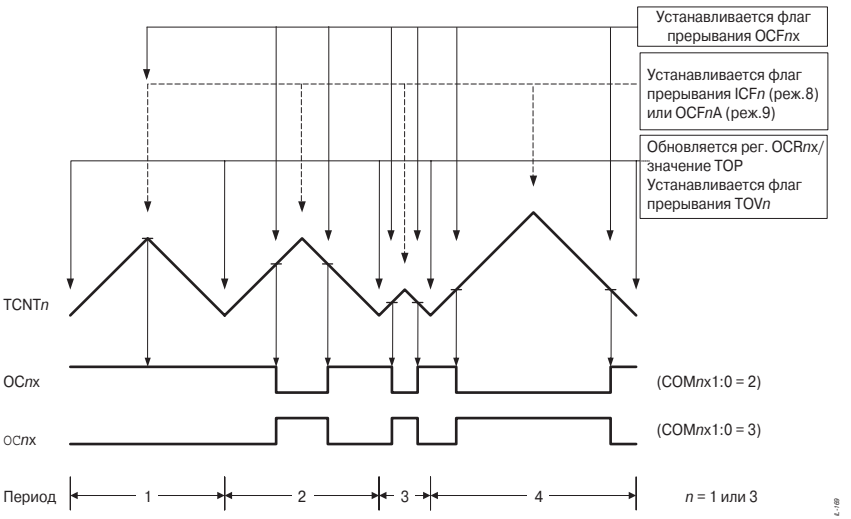


Рис. 2.83. Формирование ШИМ-сигнала в режиме Phase and Frequency Correct PWM

Если сравнить **Рис. 2.82** и **Рис. 2.83**, можно увидеть, что в режиме Phase and Frequency Correct PWM каждый период сигнала является полностью симметричным. Это является следствием 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 генерируемого сигнала остается постоянной. Частота генерируемого сигнала определяется выражением $f_{OCn} = f_{clk\_I/O}/(2N \cdot TOP)$, где TOP — модуль счета, а N — коэффициент деления предделителя (см. **Табл. 2.80**).

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. 2.103**.

Таблица 2.103. **Устойчивые состояния выхода схемы сравнения**

COM*n*A1/COM*n*B1/ COM*n*C1	**COM*n*A0/COM*n*B0/ COM*n*C0**	**Регистр OCR*n*A/ OCR*n*B/OCR*n*C**	**Состояние вывода OC*n*A/OC*n*B/ OC*n*C**
1	0	$0000	0
1	0	*TOP*	1
1	1	$0000	1
1	1	*TOP*	0

Примечание: $n = 1$ или 3.

13.7. Сторожевой таймер

Как и микроконтроллеры остальных семейств, все микроконтроллеры семейства Mega имеют в своем составе сторожевой таймер, предназначенный для защиты микроконтроллера от сбоев в процессе работы. Исполнение сторожевого таймера, структурная схема которого приведена на **Рис. 2.84**, одинаково для всех моделей семейства.

Сторожевой таймер 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 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, равные его периоду, он выполняет сброс микроконтроллера. Чтобы

избежать сброса при нормальном выполнении программы, сторожевой таймер необходимо регулярно сбрасывать через промежутки времени, меньшие его периода. Сброс сторожевого таймера осуществляется командой WDR.

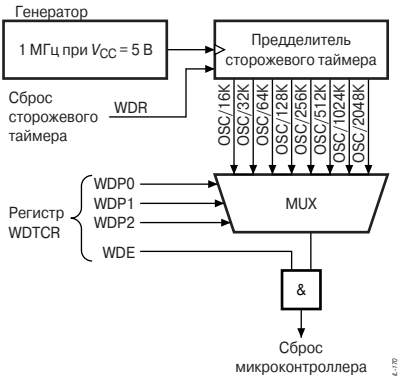


Рис. 2.84. Структурная схема сторожевого таймера

Для управления сторожевым таймером предназначен регистр WDTCR, который во всех моделях расположен по адресу $21 ($41). Формат этого регистра приведен на **Рис. 2.85**, а описание его разрядов — в **Табл. 2.104**.

	7	6	5	4	3	2	1	0	
	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0	ATmega8x ATmega8515x ATmega162x
Чтение (R)/Запись (W)	R	R	R	R/W	R/W	R/W	R/W	R/W	ATmega64x
Начальное значение	0	0	0	0	0	0	0	0	ATmega128x

	7	6	5	4	3	2	1	0	
	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	ATmega16x ATmega161x ATmega163x
Чтение (R)/Запись (W)	R	R	R	R/W	R/W	R/W	R/W	R/W	ATmega32x
Начальное значение	0	0	0	0	0	0	0	0	ATmega323x

Рис. 2.85. Формат регистра WDTCR

Таблица 2.104. **Разряды регистра WDTCR**

Разряд	Название	Описание
7...5	—	Зарезервировано, читается как «0»
4	WDTOE (WDCE*)	Разрешение изменения состояния сторожевого таймера (Разрешение выключения сторожевого таймера)

Продолжение таблицы 2.104

Разряд	Название	Описание
3	WDE	Разрешение сторожевого таймера («1» — включен)
2	WDP2	Коэффициент деления предделителя сторожевого таймера
1	WDP1	
0	WDP0	
* В моделях ATmega8x, ATmega8515x, ATmega162x, ATmega64x, ATmega128x.		

Для включения/выключения сторожевого таймера используются два разряда регистра WDTCR — WDE и WDTOE (WDCE в ряде моделей). Если разряд WDE установлен в «1», сторожевой таймер включен, если сброшен в «0» — выключен. Непосредственно перед включением таймера рекомендуется выполнять его сброс командой WDR.

Для выключения сторожевого таймера, а в некоторых случаях и для изменения периода тайм-аута необходимо выполнить определенную последовательность команд. Это сделано для того, чтобы свести к минимуму вероятность непреднамеренного изменения конфигурации сторожевого таймера. Теперь рассмотрим подробнее конфигурирование сторожевого таймера в различных моделях семейства.

ATmega16x, ATmega161x, ATmega163x, ATmega32x, ATmega323x

В этих моделях для выключения сторожевого таймера необходимо одной командой записать лог. 1 в разряды WDE и WDTOE, в течение следующих четырех машинных циклов записать лог. 0 в разряд WDE.

Изменение периода тайм-аута может быть выполнено в любой момент времени без всяких ограничений.

ATmega8x, ATmega8515x, ATmega162x, ATmega64x, ATmega128x

В этих моделях предусмотрено несколько так называемых «уровней безопасности», каждый из которых накладывает определенные ограничения на изменение конфигурации сторожевого таймера. Для выбора конкретного уровня почти во всех рассматриваемых моделях, кроме ATmega8x, используются две конфигурационные ячейки. Первая ячейка, общая для всех моделей, называется WDTON. Второй ячейкой, определяющей уровень безопасности, служит ячейка, которая переводит микро-

контроллер в режим совместимости с какими-либо другими моделями. Соответственно в моделях ATmega8515x это ячейка S8515C, в моделях ATmega162x — M161C, а в моделях ATmega64x/128x — M103C. В моделях ATmega8x такой ячейки нет. Соответствие между состоянием этих ячеек и конфигурацией сторожевого таймера приведено в **Табл. 2.105**.

Таблица 2.105. **Конфигурация сторожевого таймера**

S8515C (ATmega8515x) M161C (ATmega162x) M103C (ATmega64x/128x)	WDTON	Уровень	Начальное состояние сторожевого таймера	Выключение сторожевого таймера	Изменение периода тайм-аута
1	1	1	Выключен	Последовательность команд	Последовательность команд
1	0	2	Включен	Всегда включен	Последовательность команд
0	1	0	Выключен	Последовательность команд	Без ограничений
0	0	2	Включен	Всегда включен	Последовательность команд

Режимы управления сторожевыми таймерами характеризуются тремя уровнями: 0, 1 и 2.

В режиме уровня 0 управление сторожевым таймером осуществляется как в более ранних моделях (см. описание предыдущей группы микроконтроллеров). При включении микроконтроллера сторожевой таймер выключен, однако он может быть включен в любой момент записью лог. 1 в разряд WDE регистра WDTCR. В моделях ATmega8x этот уровень не реализован.

Уровень 1 — это когда для выключения сторожевого таймера или для изменения периода тайм-аута необходимо одной командой записать лог. 1 в разряды WDE и WDTCE, а в течение следующих четырех машинных циклов записать тоже одной командой требуемые значения в разряды WDE и/или WDP2...0, одновременно сбрасывая разряд WDCE.

В режиме уровня 2 сторожевой таймер включен постоянно (разряд WDE всегда читается как «1») и не может быть выключен. Для изменения периода тайм-аута необходимо одной командой записать лог. 1 в разряды WDE и WDTCE, а в течение следующих четырех машинных циклов записать требуемое значение в разряды WDP2...0, одновременно сбрасывая разряд WDCE. Значение, записываемое в разряд WDE, безразлично.

Период тайм-аута сторожевого таймера задается с помощью разрядов WDP2...WDP0 регистра WDTCR согласно **Табл. 2.106**.

Таблица 2.106. **Задание периода сторожевого таймера**

WDP2	WDP1	WDP0	Число тактов генератора	Период наступления тайм-аута (типовое значение) [мс]	
				V_{CC} = 3.0 В	V_{CC} = 5.0 В
0	0	0	16K (16384)	17	16
0	0	1	32K (32768)	34	33
0	1	0	64K (65536)	69 с	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

Чтобы избежать непреднамеренного сброса микроконтроллера при изменении периода сторожевого таймера, необходимо перед записью разрядов WDP2...WDP0 либо запретить сторожевой таймер, либо сбросить его.

Глава 14. **Аналоговый компаратор**

14.1. Введение

Модуль аналогового компаратора входит в состав всех без исключения микроконтроллеров семейства Mega. Будучи включенным, компаратор позволяет сравнивать значения напряжений, присутствующих на двух выводах микроконтроллера. Результатом сравнения является логическое значение, которое может быть прочитано из программы. По результату сравнения может быть сгенерировано прерывание, а также осуществлен захват состояния таймера/счетчика T1. Последняя функция позволяет, в частности, измерять длительности аналоговых сигналов.

Используемые компаратором выводы являются контактами портов ввода/вывода общего назначения (см. **Табл. 2.107**).

Таблица 2.107. **Выводы, используемые аналоговым компаратором**

Название	**ATmega8x**	**ATmega8515x**	**ATmega16x**	**ATmega161x**	**ATmega162x**	**ATmega163x**	**ATmega323x**	**ATmega32x**	**ATmega323x**	**ATmega64x ATmega128x**	**Назначение**
AIN0	PD6	PB2	PB2	PB2	PB2	PB2	PB2	PB2	PB2	PE2	Неинвертирующий вход
AIN1	PD7	PB3	PB3	PB3	PB3	PB3	PB3	PB3	PB3	PE3	Инвертирующий вход

Чтобы указанные выводы могли использоваться аналоговым компаратором, они должны быть сконфигурированы как входы (соответствующий разряд регистра DDR*x* установлен в «1»). Кроме того, необходимо отключить внутренние подтягивающие резисторы записью лог. 0 в соответствующий разряд регистра PORT*x*.

14.2. Функционирование компаратора

Структурная схема аналогового компаратора приведена на **Рис. 2.86**.

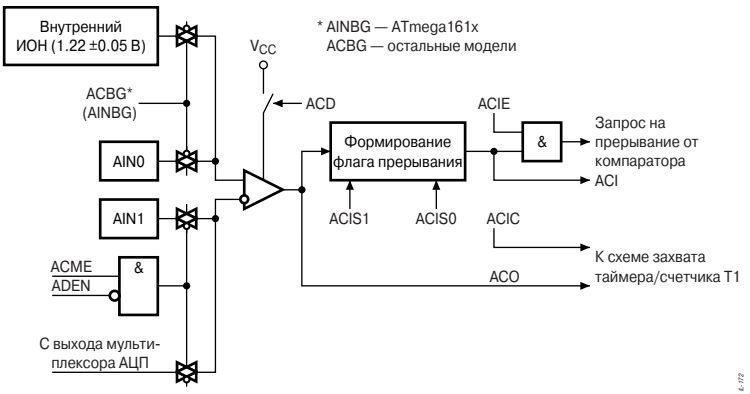


Рис. 2.86. Структурная схема аналогового компаратора

Управление компаратором и контроль его состояния осуществляется с помощью регистра ACSR, который во всех моделях расположен по адресу $08 ($28). Формат этого регистра приведен на **Рис. 2.87**, а назначение его разрядов кратко описано в **Табл. 2.108**.

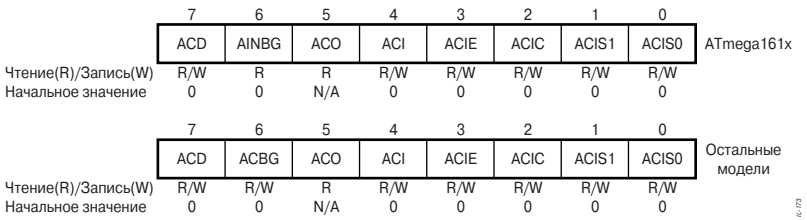


Рис. 2.87. Формат регистра ACSR

Таблица 2.108. **Разряды регистра ACSR**

Разряд	Название	Описание
7	ACD	Выключение компаратора («0» — включен, «1» — выключен)
6	ACBG AINBG*	Подключение к неинвертирующему входу компаратора внутреннего ИОН («0» — не подключен, «1» — подключен)
5	ACO	Результат сравнения (выход компаратора)

Продолжение таблицы 2.108

Разряд	Название	Описание
4	ACI	Флаг прерывания от компаратора
3	ACIE	Разрешение прерывания от компаратора
2	ACIC	Подключение компаратора к схеме захвата таймера/счетчика T1 («1» — подключен, «0» — отключен)
1, 0	ACIS1:ACIS0	Условие возникновения прерывания от компаратора
* В моделях ATmega161x.		

По своему действию рассматриваемый узел микроконтроллера является обычным компаратором. Если напряжение на выводе AIN0 (неинвертирующий вход) больше напряжения на выводе AIN1 (инвертирующий вход), то результат сравнения будет равен «1». В противном случае результат сравнения будет равен «0». Этот результат (состояние выхода компаратора) сохраняется в разряде ACO регистра ACSR.

Разряд ACD отвечает за включение и выключение компаратора. Поскольку при подаче напряжения питания все разряды регистра ACSR сбрасываются в «0», компаратор включается автоматически при включении микроконтроллера. Для выключения компаратора разряд ACD следует установить в «1». При изменении состояния этого разряда прерывание от компаратора следует запретить.

Как уже было сказано, по результату сравнения схема компаратора может генерировать запрос на прерывание. Если состояние выхода компаратора (разряд ACO) изменилось заданным образом, устанавливается флаг прерывания ACI регистра ACSR и генерируется запрос на прерывание. Как и для других прерываний, этот флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания или программно, записью в него лог. 1. Для разрешения прерывания необходимо установить в «1» разряд ACIE регистра ACSR и, разумеется, флаг I регистра SREG.

Какое именно изменение состояния выхода компаратора вызовет прерывание, определяется состоянием разрядов ACIS1:ACIS0 регистра ACSR согласно **Табл. 2.109**. При изменении этих разрядов прерывание от компаратора (как и для разряда ACD) должно быть запрещено.

Таблица 2.109. **Условия генерации запроса на прерывание от компаратора**

ACIS1	ACIS0	Условие
0	0	Любое изменение состояния выхода компаратора
0	1	Зарезервировано

Продолжение таблицы 2.109

ACIS1	ACIS0	Условие
1	0	Изменение состояния выхода компаратора с «1» на «0»
1	1	Изменение состояния выхода компаратора с «0» на «1»

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

Компаратор может сравнивать сигналы, присутствующие не только на выводах AIN0 и AIN1. Так, вместо вывода AIN0 микроконтроллера к неинвертирующему входу компаратора может быть подключен внутренний источник опорного напряжения (ИОН) величиной 1.22 ±0.1 В. Для этого необходимо установить в «1» разряд ACBG (AINBG в моделях ATmega161x) регистра ACSR. Причем если схема BOD выключена (BODEN = «1»), то между подключением ИОН к компаратору и началом использования компаратора необходимо выждать некоторое время (не более 100 мкс).

На инвертирующий вход компаратора может также поступать сигнал с выхода мультиплексора модуля АЦП. Другими словами, вместо вывода AIN1 микроконтроллера инвертирующий вход компаратора может быть подключен к любому из входов АЦП ADC0...ADC7 (ADC0...ADC5 в моделях ATmega8x). Естественно, в моделях ATmega8515x, ATmega161x и ATmega162x, не имеющих модуля АЦП, эта возможность недоступна.

Подключение выхода мультиплексора АЦП к входу компаратора осуществляется установкой в «1» разряда ACME регистра специальных функций SFIOR (3-й разряд регистра). Разумеется, модуль АЦП при этом должен быть выключен (разряд ADEN регистра ADCSRA сброшен в «0»). Какой именно из входов АЦП будет использоваться в качестве инвертирующего входа компаратора, определяется разрядами MUX2...0 регистра ADMUX, как показано в **Табл. 2.110**.

Таблица 2.110. **Управление инвертрующим входом компаратора**

ACME	ADEN	MUX2…0	Инвертирующий вход компаратора
0	x	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1

Продолжение таблицы 2.110

ACME	ADEN	MUX2...0	Инвертирующий вход компаратора
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6*
1	0	111	ADC7*
* В моделях ATmega8x отсутствуют.			

Подробно регистры ADCSR и ADMUX будут рассмотрены в следующем разделе, посвященному модулю АЦП.

В заключение ознакомьтесь с **Табл. 2.111**, в которой представлены основные параметры аналогового компаратора.

Таблица 2.111. **Основные параметры аналогового компаратора**

Обозначение	Параметр	Условия	min	max
V_{ACIO}	Входное напряжение смещения [мВ]	V_{CC} = 5 В, V_{IN} = V_{CC}/2	—	40.0
I_{ACLK}	Ток утечки на входе [нА]	V_{CC} = 5 В, V_{IN} = V_{CC}/2	–50.0	50.0
t_{ACPD}	Время отклика [нс]	V_{CC} = 2.7 В V_{CC} = 4.0 В	—	750 500

Глава 15. **Аналого-цифровой преобразователь**

15.1. Общие сведения

Модуль 10-разрядного АЦП последовательного приближения входит в состав моделей ATmega8x, ATmega16x, ATmega163x, ATmega32x, ATmega323x, ATmega64x и ATmega128x. Основные параметры этого АЦП следующие:

- абсолютная погрешность: ±2 МЗР;
- интегральная нелинейность: ±0.5 МЗР;
- быстродействие: до 15 тыс. выборок/с.

На входе модуля АЦП всех моделей имеется 8-канальный аналоговый мультиплексор, предоставляющий в распоряжение пользователя 8 каналов с несимметричными входами. Отдельно следует сказать о микроконтроллерах ATmega8x. Микроконтроллеры этой группы, выпускаемые в корпусе DIP-32, имеют только 6 каналов преобразования. Кроме того, во всех моделях ATmega8x два канала (ADC4 и ADC5) являются 8-разрядными.

В моделях ATmega16x, ATmega32x, ATmega64x, Atmega128x входы АЦП могут также объединяться попарно для формирования в общей сложности до 13 каналов с дифференциальным входом. Два канала при этом имеют возможность 20- и 200-кратного предварительного усиления входного сигнала. При коэффициентах усиления 1x и 20x действительная разрешающая способность составляет 8 разрядов, а при коэффициенте усиления 200x — 7 разрядов.

В качестве источника опорного напряжения для АЦП может использоваться как напряжение питания микроконтроллера, так и внутренний либо внешний источник опорного напряжения.

В процессе работы АЦП может функционировать в двух режимах:

- режим одиночного преобразования, когда запуск каждого преобразования инициируется пользователем;
- режим непрерывного преобразования, когда запуск преобразований выполняется непрерывно через определенные интервалы времени.

15.2. Функционирование модуля АЦП

Структурная схема модуля АЦП приведена на **Рис. 2.89**. Элементы схемы, выделенные на рисунке серым цветом, и связанные с ними сигналы присутствуют только в моделях ATmega16x, ATmega32x, ATmega64x, ATmega128x. В остальных моделях неинвертирующий вход компаратора выборки/хранения подключен непосредственно к выходу мультиплексора (показано пунктиром).

Регистры, используемые для управления модулем АЦП в различных моделях, приведены в **Табл. 2.112**.

Таблица 2.112. **Регистры управления модулем АЦП**

Регистр	Адрес	ATmega8x	ATmega16x	ATmega163x	ATmega323x	ATmega32x	ATmega64x	ATmega128x	Описание
ADCSR	$06 ($26)	♦	—	♦	♦	—	—	—	Регистр управления и состояния
ADCSRA	$06 ($26)	—	♦	—	—	♦	♦	♦	Регистр управления и состояния A
ADCSRB	($8E)	—	—	—	—	—	♦	—	Регистр управления и состояния B
ADMUX	$07 ($27)	♦	♦	♦	♦	♦	♦	♦	Регистр управления мультиплексором
SFIOR	$30 ($50)	♦	♦	—	—	♦	—	—	Регистр специальных функций
	$20 ($40)	—	—	—	—	—	♦	♦	

Формат регистров ADCSRA (ADCSR) и ADMUX приведен на **Рис. 2.88** и **Рис. 2.90**, а краткое описание функций их разрядов приведено в **Табл. 2.113** и **Табл. 2.114** соответственно.

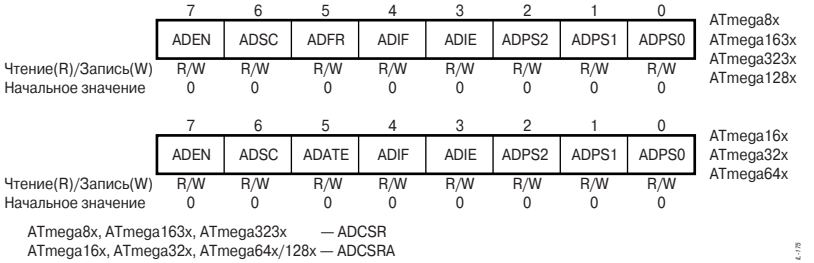


Рис. 2.88. Формат регистра ADCSRA (ADCSR)

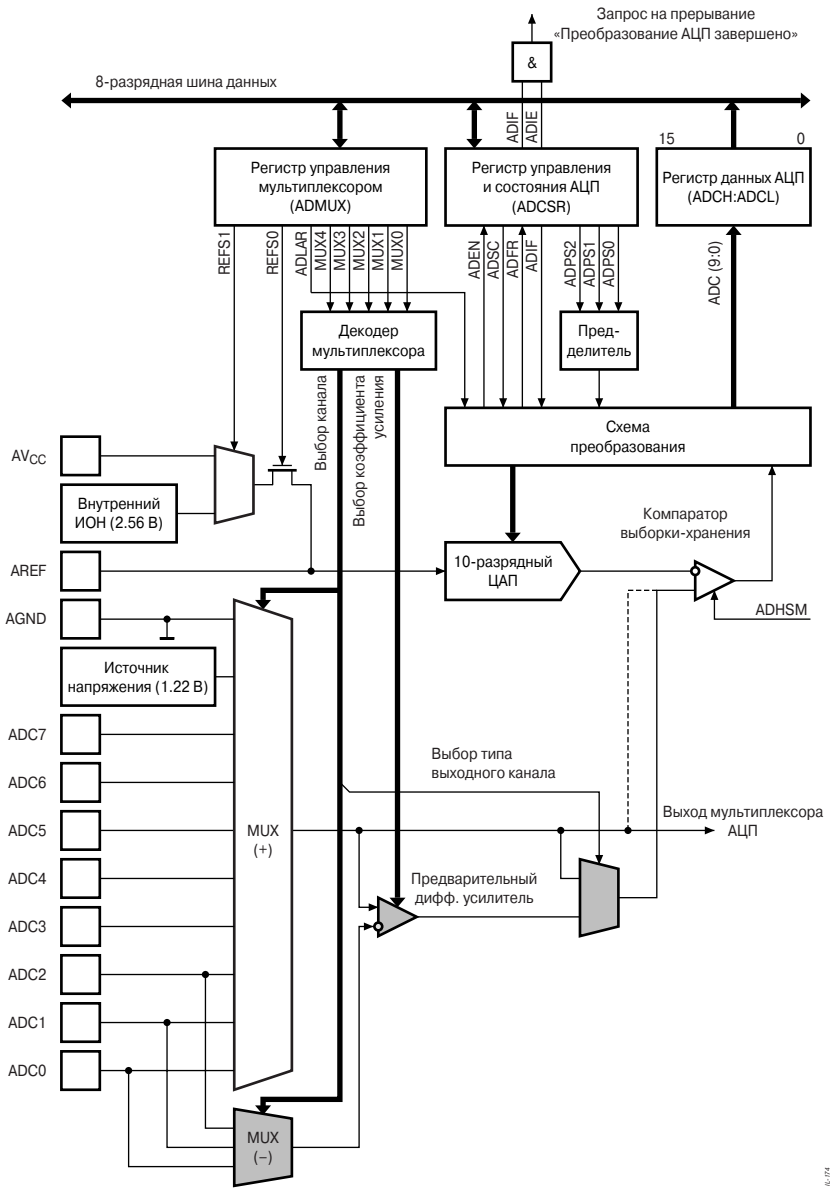


Рис. 2.89. Структурная схема модуля АЦП

Таблица 2.113. **Разряды регистра ADCSRA (ADCSR*)**

Разряд	Название	Описание
7	ADEN	Разрешение АЦП (1 — включено, 0 — выключено)
6	ADSC	Запуск преобразования (1 — начать преобразование)
5	ADFR (ADATE**)	Выбор режима работы АЦП
4	ADIF	Флаг прерывания от компаратора
3	ADIE	Разрешение прерывания от компаратора
2...0	ADPS2:ADPS0	Выбор частоты преобразования
* В моделях ATmega8x, ATmega163x и ATmega323x. ** В моделях ATmega16x, ATmega32x и ATmega64x.		

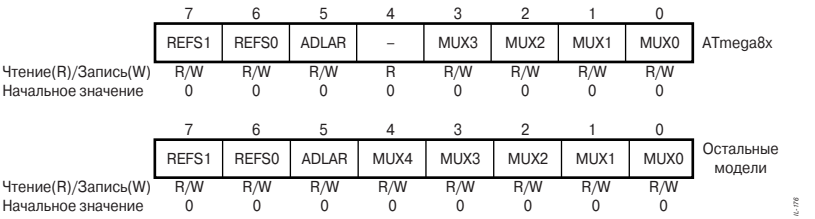


Рис. 2.90. Формат регистра ADMUX

Таблица 2.114. **Разряды регистра ADMUX**

Разряд	Назввание	Описание	Модель
7, 6	REFS1:REFS0	Выбор источника опорного напряжения	Все модели
5	ADLAR	Выравнивание результата преобразования	Все модели
4	—	Зарезервировано	ATmega8x
	MUX4	Выбор входного канала	Остальные
3...0	MUX3...MUX0	Выбор входного канала	Все модели

Формат регистров ADCSRB и SFIOR приведен на **Рис. 2.91** и **Рис. 2.92** соответственно (не используемые в данном случае разряды регистра SFIOR указаны на рисунке как «X»).

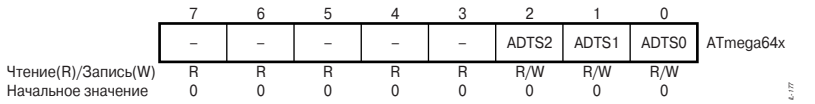


Рис. 2.91. Формат регистра ADCSRB

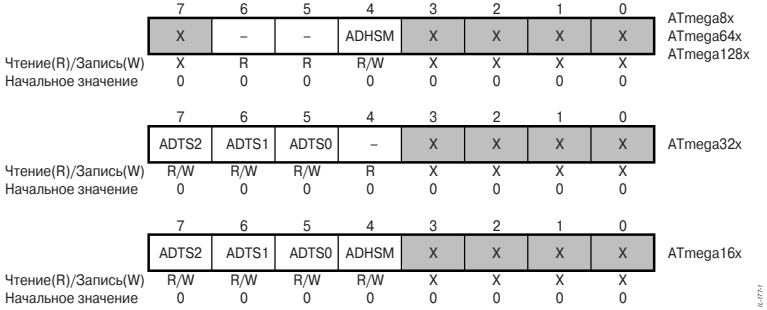


Рис. 2.92. Регистр SFIOR

Для разрешения работы АЦП необходимо записать лог. 1 в разряд ADEN регистра ADCSR, а для выключения — лог. 0. Если АЦП будет выключено во время цикла преобразования, то преобразование завершено не будет (в регистре данных АЦП останется результат предыдущего преобразования).

В моделях ATmega8x, ATmega163x, ATmega323x и ATmega128x режим работы АЦП определяется состоянием разряда ADFR. Если он установлен в «1», АЦП работает в режиме непрерывного преобразования. В этом режиме запуск каждого следующего преобразования осуществляется автоматически после окончания текущего. Если же разряд ADFR сброшен в «0», АЦП работает в режиме одиночного преобразования и запуск каждого преобразования осуществляется по команде пользователя.

В микроконтроллерах ATmega16x, ATmega32x и ATmega64x запуск АЦП возможен не только по команде пользователя, но и по прерыванию от некоторых периферийных устройств, имеющихся в составе микроконтроллера. Для выбора режима работы в этих моделях используется разряд ADATE регистра ADCSR и разряды ADTDS2...0 регистра SFIOR (ADCSRB в моделях ATmega64x).

Если разряд ADATE сброшен в «0», АЦП работает в режиме одиночного преобразования. Если же разряд ADTAE установлен в «1», функционирование АЦП определяется содержимым разрядов ADTS2...0 согласно **Табл. 2.115**.

Таблица 2.115. **Источник сигнала для запуска преобразования в ATmega16x**

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

Продолжение таблицы 2.115

ADTS2	ADTS1	ADTS0	Источник стартового сигнала
1	0	1	Прерывание по событию «Совпадение B» таймера/счетчика T1
1	1	0	Прерывание по переполнению таймера/счетчика T1
1	1	1	Прерывание по событию «Захват» таймера/счетчика T1

Запуск каждого преобразования в режиме одиночного преобразования, а также запуск первого преобразования в режиме непрерывного преобразования осуществляется установкой в «1» разряда ADSC регистра ADCSR. Запуск преобразования по прерыванию (модели ATmega16x, ATmega32x и ATmega64x) осуществляется при установке в «1» флага выбранного прерывания. Разряд ADSC регистра ADCSR при этом аппаратно устанавливается в «1». Запуск преобразования в этих режимах также может быть осуществлен установкой в «1» разряда ADSC регистра ADCSR.

В режимах одиночного и непрерывного преобразований цикл преобразования начинается по первому нарастающему фронту тактового сигнала после установки разряда ADSC. Если используется запуск по прерыванию, то цикл преобразования начинается по первому нарастающему фронту тактового сигнала после установки флага выбранного прерывания. Причем при установке этого флага осуществляется сброс предделителя модуля АЦП. Тем самым обеспечивается фиксированная задержка между генерацией запроса на прерывание и началом цикла преобразования.

Длительность цикла составляет 13 тактов при использовании несимметричного входа и 13 либо 14 тактов при использовании дифференциального входа; выборка и запоминание входного сигнала осуществляется в течение первых 1.5 и 2.5 тактов соответственно. Через 13 (14) тактов преобразование завершается, разряд ADSC аппаратно сбрасывается в «0» (в режиме одиночного преобразования) и результат преобразования сохраняется в регистре данных АЦП. Одновременно устанавливается флаг прерывания ADIF регистра ADCSR и генерируется запрос на прерывание. Как и флаги остальных прерываний, флаг ADIF сбрасывается аппаратно при запуске подпрограммы обработки прерывания от АЦП или программно, записью в него лог. 1. Разрешение прерывания осуществляется установкой в «1» разряда ADIE регистра ADCSR при установленном флаге I регистра SREG.

Если АЦП работает в режиме непрерывного преобразования, новый цикл начнется сразу же после записи результата. В режиме одиночного преобразования новое преобразование может быть запущено сразу же после сброса разряда ADSC (до сохранения результата текущего преобразования). Однако реально цикл преобразования начнется не ранее чем через один такт после окончания текущего преобразования. Временные диаграммы, иллюстрирующие сказанное, приведены на **Рис. 2.93**.

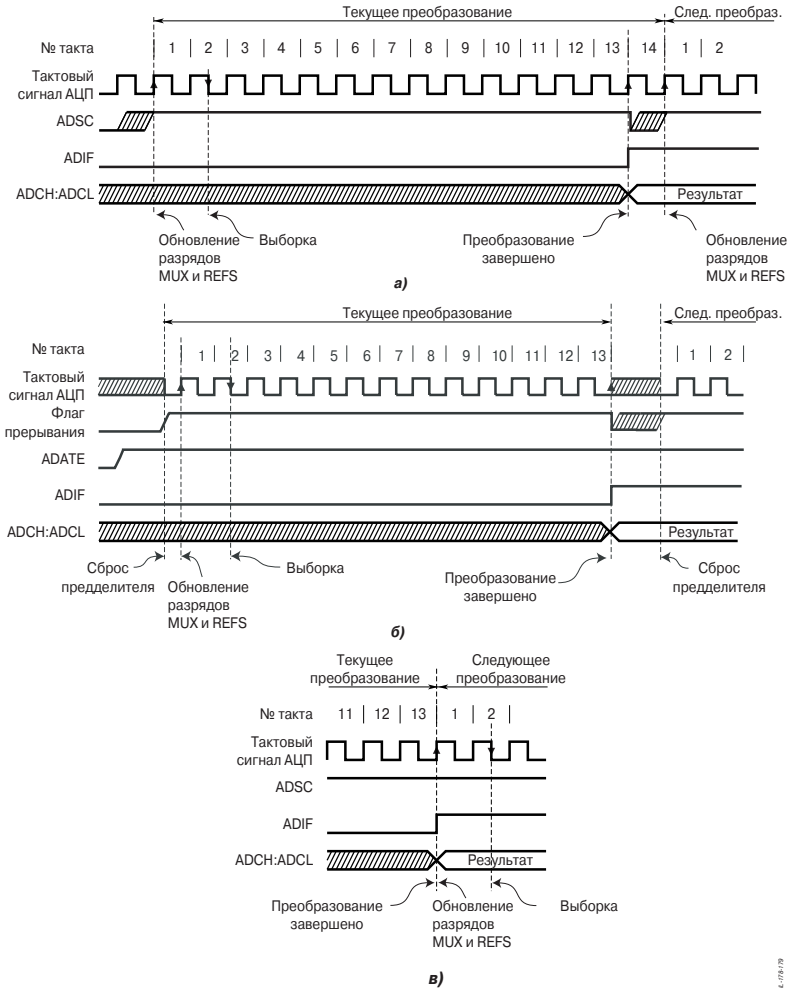


Рис. 2.93. Временные диаграммы работы АЦП в режиме одиночного преобразования (*а*), в режиме запуска по прерыванию (*б*) и в режиме непрерывного преобразования (*в*)

При запуске первого преобразования после включения АЦП для выполнения преобразования потребуется 25 тактов, т. е. на 12 тактов больше, чем обычно. В течение этих 12 тактов выполняется «холостое» преобразование, инициализирующее АЦП (**Рис. 2.94**).

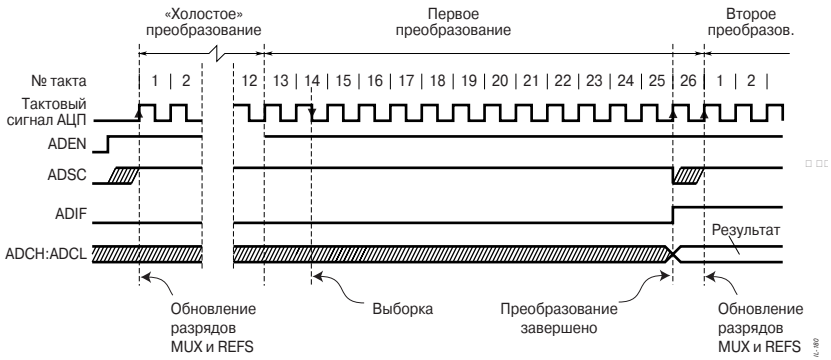


Рис. 2.94. Временные диаграммы работы АЦП при первом преобразовании (режим одиночного преобразования)

Для формирования тактовой частоты модуля АЦП в нем имеется отдельный предделитель. Коэффициент деления предделителя и соответственно длительность преобразования определяется состоянием разрядов ADPS2...ADPS0 регистра ADCSR (см. **Табл. 2.116**).

Таблица 2.116. **Задание коэффициента деления предделителя АЦП**

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

Наибольшая точность преобразования достигается, если тактовая частота модуля АЦП находится в диапазоне 50...200 кГц. Соответственно коэффициент деления предделителя рекомендуется выбирать таким, чтобы тактовая частота модуля АЦП находилась в указанном диапазоне. Если же точности преобразования меньше 10 разрядов достаточно, можно использовать более высокую частоту, увеличивая тем самым частоту выборки. В моделях ATmega8x, ATmega16x, ATmega64x и ATmega128x для этой же цели предназначен разряд ADCHM регистра SFIOR. При установке этого разряда в «1» скорость преобразования АЦП увеличивается. Однако при этом увеличивается и потребление микроконтроллера.

В моделях ATmega8x выводы микроконтроллера, подключенные к входу АЦП, определяются состоянием разрядов MUX3...MUX0 регистра ADMUX согласно **Табл. 2.117**. В остальных моделях микроконтроллеров для этой же цели используются разряды MUX4...MUX0 регистра (см. **Табл. 2.118**). Для каналов с дифференциальным входом указанные разряды определяют также коэффициент предварительного усиления входного сигнала.

Таблица 2.117. **Управление входным мультиплексором в моделях ATmega8x**

MUX3...MUX0	Несимметричный вход
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4*
0101	ADC5*
0110	ADC6**
0111	ADC7**
1000...1101	Зарезервировано
1110	1.22 В
1111	0 В (GND)

* 8-разрядное преобразование.
** Имеются только в корпусах TQFP-32 и MLF-32.

Таблица 2.118. **Управление входным мультиплексором в моделях ATmega16x, ATmega163x/32x/323x/64x/128x**

MUX4...MUX0	Несимметричный вход	Дифференциальный вход		Предвар. усиление
		(положительный)	(отрицательный)	
00000	ADC0	Не применимо		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			

Продолжение таблицы 2.118

MUX4...MUX0	Несимметричный вход	Дифференциальный вход		Предвар. усиление
		(положительный)	(отрицательный)	
01000*	Не применимо	ADC0	ADC0	10x
01001*		ADC1	ADC0	10x
01010*		ADC0	ADC0	200x
01011*		ADC1	ADC0	200x
01100*		ADC2	ADC2	10x
01101*		ADC3	ADC2	10x
01110*		ADC2	ADC2	200x
01111*		ADC3	ADC2	200x
10000*		ADC0	ADC1	1x
10001*		ADC1	ADC1	1x
10010*		ADC2	ADC1	1x
10011*		ADC3	ADC1	1x
10100*		ADC4	ADC1	1x
10101*		ADC5	ADC1	1x
10110*		ADC6	ADC1	1x
10111*		ADC7	ADC1	1x
11000*		ADC0	ADC2	1x
11001*		ADC1	ADC2	1x
11010*		ADC2	ADC2	1x
11011*		ADC3	ADC2	1x
11100*		ADC4	ADC2	1x
11101*		ADC5	ADC2	1x
11110	1.22 В	Не применимо		
11111	0 В (GND)			
* В моделях ATmega163x и ATmega323x эти значения разрядов зарезервированы.				

Следует отметить, что предварительный усилитель, используемый каналами с дифференциальным входом, имеет встроенную схему коррекции смещения. Оставшаяся после коррекции величина смещения может быть учтена программным способом. Для этого следует оба входа дифференциального усилителя подключить к одному и тому же выводу микроконтрол-

лера (**Табл. 2.117**), а затем вычитать полученное значение из результата последующих преобразований. Таким образом, ошибка смещения может быть снижена до величины, меньшей 1 МЗР.

Состояние разрядов MUX2...MUX0 можно изменить в любой момент, однако, если это будет сделано во время цикла преобразования, смена канала произойдет только после завершения преобразования. Благодаря этому в режиме непрерывного преобразования можно легко осуществлять последовательное преобразование сигналов нескольких каналов.

Отдельно следует сказать о каналах с дифференциальным входом. После смены таких каналов первое измерение следует производить не ранее чем через 125 мкс после выбора канала. Указанное время требуется для установления значения коэффициента усиления предусилителя. Соответственно значения, измеренные до истечения этого срока, не могут считаться достоверными.

Как уже было отмечено, модуль АЦП может использовать различные источники опорного напряжения (ИОН). Выбор конкретного источника опорного напряжения осуществляется с помощью разрядов REFS1:REFS0 регистра ADMUX (**Табл. 2.119**).

Таблица 2.119. **Выбор источника опорного напряжения**

REFS1	REFS0	Источник опорного напряжения
0	0	Внешний ИОН, подключенный к выводу AREF; внутренний ИОН отключен
0	1	Напряжение питания AV_{CC}*
1	0	Зарезервировано
1	1	Внутренний ИОН напряжением 2.56 В, подключенный к выводу AREF*
* Если к выводу AREF подключен источник напряжения, данные варианты использоваться не могут.		

Как указано в таблице, внутренний ИОН подключен к выводу AREF микроконтроллера. Поэтому при его использовании к выводу AREF можно подключить внешний фильтрующий конденсатор для повышения помехозащищенности.

15.3. Результат преобразования

После завершения преобразования (при установке в «1» флага ADIF регистра ADCSR) его результат сохраняется в регистре данных АЦП. Поскольку АЦП имеет 10 разрядов, этот регистр физически размещен в двух регистрах ввода/вывода ADCH:ADCL, доступных только для чтения. Эти

регистры расположены по адресам $05:$04 и при включении микроконтроллера содержат значение «$0000».

По умолчанию результат преобразования выравнивается вправо (старшие 6 разрядов регистра ADCH — незначащие). Однако он может выравниваться и влево (младшие 6 разрядов регистра ADCL — незначащие). Для управления выравниванием результата преобразования служит разряд ADLAR регистра ADMUX. Если этот разряд установлен в «1», результат преобразования выравнивается по левой границе 16-разрядного слова, если сброшен в «0» — по правой границе.

Обращение к регистрам ADCH и ADCL для получения результата преобразования должно выполняться в определенной последовательности: сначала необходимо прочитать регистр ADCL, а затем ADCH. Это требование связано с тем, что после обращения к регистру ADCL процессор блокирует доступ к регистрам данных со стороны АЦП до тех пор, пока не будет прочитан регистр ADCH. Благодаря этому можно быть уверенным, что при чтении регистров в них будут находиться составляющие одного и того же результата. Соответственно, если очередное преобразование завершится до обращения к регистру ADCH, результат преобразования будет потерян. С другой стороны, если результат преобразования выравнивается влево и достаточно точности 8-разрядного значения, для получения результата можно прочитать только содержимое регистра ADCH.

Для каналов с несимметричным входом результат преобразования определяется выражением $ADC = 1024V_{IN}/V_{REF}$, где V_{IN} — значение входного напряжения, а V_{REF} — величина опорного напряжения.

Для каналов с дифференциальным входом результат преобразования определяется выражением $ADC = 512K(V_{POS} - V_{NEG})/V_{REF}$, где V_{POS} – величина напряжения на положительном входе, V_{NEG} – величина напряжения на отрицательном входе, а K — коэффициент усиления.

Результат преобразования представляется в этом случае в дополнительном коде, а его значение лежит в диапазоне $200 (–512)...$1FF (+512).

15.4. Повышение точности преобразования

В этом параграфе приведены некоторые рекомендации, позволяющие в наибольшей степени использовать возможности АЦП. Прежде всего, для минимизации погрешности самого АЦП необходимо правильно выбрать тактовую частоту преобразования. С этой же целью на входе АЦП рекомендуется устанавливать фильтр низких частот. Кроме того, при разработке конструкции и топологии печатной платы следует придерживаться общих правил проектирования цифроаналоговых устройств:

- на печатной плате необходимо предусмотреть область сплошной металлизации под аналоговую «землю». Аналоговая часть микроконтроллера и аналоговая часть всего устройства должны располагаться над этой областью. Аналоговая и цифровая «земли» должны соединяться друг с другом в единственной точке печатной платы;
- проводники, по которым распространяются аналоговые сигналы, должны быть как можно короче и располагаться над аналоговой «землей». Кроме того, они должны быть размещены как можно дальше от быстродействующих цифровых цепей;
- вывод AV_{CC} микроконтроллера должен подключаться к источнику питания V_{CC} через *LC*-фильтр, как показано на **Рис. 2.95** (расположение выводов показано условно);
- если какие-либо выводы АЦП используются как цифровые выходы, они не должны переключаться во время преобразования.

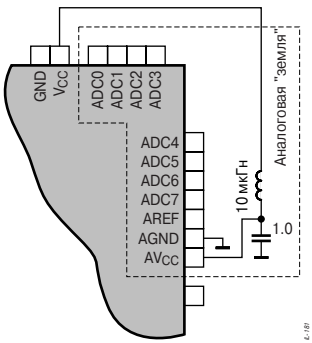


Рис. 2.95. Подключение цепей питания АЦП

Для сведения к минимуму электромагнитных помех, наводимых ядром процессора, во всех рассматриваемых микроконтроллерах имеется дополнительный «спящий» режим — ADC Noise Reduction (режим снижения шумов АЦП). В этом режиме из всех периферийных устройств функционируют только АЦП и сторожевой таймер. Для той же цели (но с меньшим эффектом) может быть использован режим Idle. Для использования АЦП в любом из указанных режимов необходимо убедиться, что АЦП включено и не занято преобразованием, затем переключить АЦП в режим одиночного преобразования и разрешить прерывание от АЦП, после чего перевести микроконтроллер в режим ADC Noise Reduction (или режим Idle).

Сразу же после остановки процессора начнется цикл преобразования.

При завершении преобразования будет сгенерировано прерывание от АЦП, которое переведет микроконтроллер в рабочий режим, и начнется выполнение подпрограммы обработки этого прерывания.

15.5. Параметры АЦП

Основные параметры АЦП приведены в **Табл. 2.120**. Все значения указаны для диапазона температур окружающей среды –40...+80°C.

Таблица 2.120. **Основные параметры АЦП**

Обозначение	Параметр	Условия	min	typ	max
	Разрешение [бит]	Несимметричный вход	—	10	—
		Дифференциальный вход, K_U = 1x и 20x	—	8	—
		Дифференциальный вход, K_U = 200x	—	7	—
	Абсолютная погрешность [МЗР]	Несимметричный вход, V_{REF} = 4 В f_{ADC} = 200 кГц, ADHSM = 0	—	1	2
		Несимметричный вход, V_{REF} = 4 В f_{ADC} = 1 МГц, ADHSM = 1	—	4	—
INL	Интегральная нелинейность [МЗР]	V_{REF} = 4 В	—	0.5	—
DNL	Дифференциальная нелинейность [МЗР]	V_{REF} = 4 В	—	0.5	—
—	Ошибка смещения [МЗР]	V_{REF} = 4 В	—	1	—
—	Время преобразования [мкс]	Режим непрерывного преобразования	65	—	260
f_{ADC}	Тактовая частота [кГц]	—	50	—	200
AV_{CC}	Напряжение питания [В]	—	V_{CC} – 0.3	—	V_{CC} + 0.3
V_{REF}	Опорное напряжение [В]	Несимметричный вход	2.0	—	V_{CC}
		Дифференциальный вход	2.0	—	V_{CC} – 0.2
V_{INT}	Напряжение внутреннего ИОН [В]	—	2.4	2.56	2.7
R_{REF}	Входное сопротивление канала опорного напряжения [кОм]	—	6	10	13
R_{AIN}	Входное сопротивление аналогового входа [МОм]	—	—	100	—

Глава 16. **Универсальный асинхронный (синхронный/асинхронный) приемопередатчик**

16.1. Общие сведения

Все без исключения микроконтроллеры семейства Mega имеют в своем составе модули либо универсального асинхронного (UART), либо универсального синхронно/асинхронного (USART) приемопередатчика. Более того, в некоторых моделях имеется по два таких модуля. Какие именно модули реализованы в отдельных микроконтроллерах можно определить по **Табл. 2.121**.

Таблица 2.121. **Модули USART/UART в микроконтроллерах семейства Mega**

Модуль приемопередатчика		ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega32x	ATmega323x	ATmega64x	ATmega128x
USART	USART0	♦	♦	♦	–	♦	–	♦	♦	♦	♦
	USART1	–	–	–	–	♦	–	–	–	♦	♦
UART	UART0	–	–	–	♦	–	♦	–	–	–	–
	UART1	–	–	–	♦	–	–	–	–	–	–

Заметим, что модули USART при работе в асинхронном режиме совместимы с модулями UART как по расположению разрядов управляющих регистров, так и по функционированию. Небольшие различия имеются только в работе схемы буферизации блока приемника модулей и в названии (но не в назначении) некоторых разрядов управляющих регистров.

Все модули приемопередатчиков обеспечивают полнодуплексный обмен по последовательному каналу, при этом скорость передачи данных может варьироваться в довольно широких пределах. В модулях UART посылка мо-

жет быть 8- или 9-разрядной, а в модулях USART ее длина может составлять от 5 до 9 разрядов. Еще одной особенностью модулей USART является наличие схем формирования и контроля четности.

Модули USART/UART, реализованные в микроконтроллерах семейства, могут обнаруживать следующие внештатные ситуации:

- переполнение;
- ошибка кадрирования;
- неверный старт-бит.

Для уменьшения вероятности сбоев в модулях также реализована такая полезная функция, как фильтрация помех.

Для взаимодействия с программой в модулях предусмотрены 3 прерывания, запрос на генерацию которых формируется при наступлении следующих событий: «передача завершена», «регистр данных передатчика пуст» и «прием завершен».

Как обычно, выводы микроконтроллера, используемые модулями USART/UART, являются линиями портов ввода/вывода общего назначения. Все выводы микроконтроллеров, используемые модулями, сведены в **Табл. 2.122**. Там же указаны функции этих выводов.

Таблица 2.122. **Выводы, используемые модулями USART/UART**

Название	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega32x	ATmega323x	ATmega64x	ATmega128x	Описание
RXD	PD0	PD0	PD0	—	—	PD0	PD0	PD0	—	—	Вход USART0 (UART0)
RXD0	—	—	—	PD0	PD0	—	—	—	PE0	PE0	
TXD	PD1	PD1	PD1	—	—	PD1	PD1	PD1	—	—	Выход USART0 (UART0)
TXD0	—	—	—	PD1	PD1	—	—	—	PE1	PE1	
XCK	PD4	PD4	PB0	—	PD4	—	PB0	PB0	PE2	PE2	Вход/выход внешнего тактового сигнала USART0
XCK0	—	—	—	—		—	—	—	—	—	
RXD1	—	—	—	PB2	PB2	—	—	—	PD2	PD2	Вход USART1 (UART1)
TXD1	—	—	—	PB3	PB3	—	—	—	PD3	PD3	Выход USART1 (UART1)
XCK1	—	—	—	—	PD2	—	—	—	PD5	PD5	Вход/выход внешнего тактового сигнала USART1

16.2. Использование модулей USART/UART

Упрощенная структурная схема одного модуля USART/UART приведена на **Рис. 2.96**. Элементы схемы, выделенные на рисунке серым цветом, имеются только в составе модулей USART.

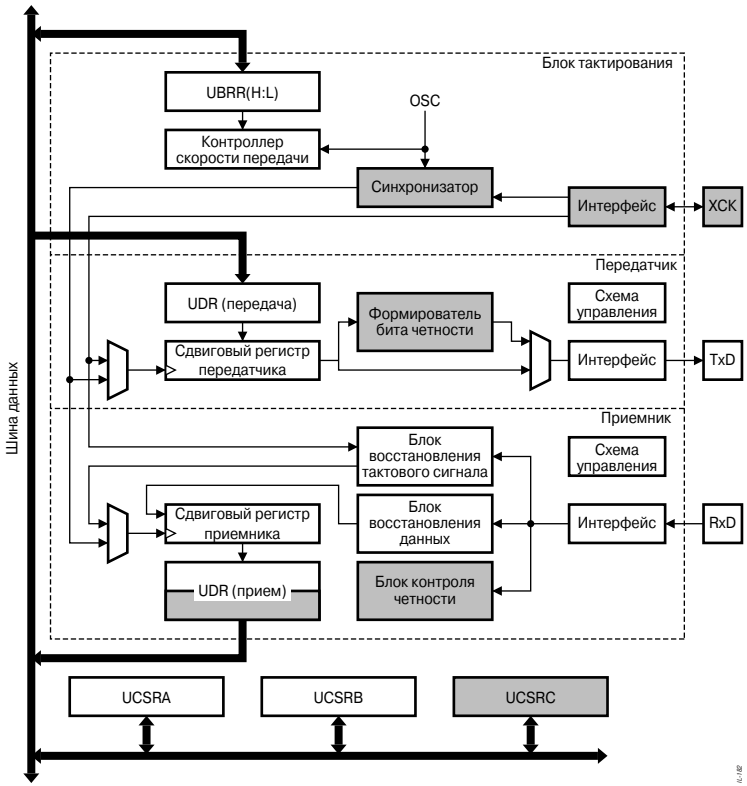


Рис. 2.96. Структурная схема модуля USART/UART

Как показано на рисунке, модуль состоит из трех основных частей: блока тактирования, блока передатчика и блока приемника. Блок тактирования модулей USART включает в себя схему синхронизации, которая используется при работе в синхронном режиме и контроллер скорости передачи. В модулях UART блок тактирования состоит только из контроллера скорости передачи.

Блок передатчика включает одноуровневый буфер, сдвиговый регистр, схему формирования бита четности (только USART) и схему управления. Блок приемника, в свою очередь, включает схемы восстановления тактового сигнала и данных, схему контроля четности (только USART), двухуровневый (USART) или одноуровневый (UART) буфер, сдвиговый регистр, а также схему управления.

Буферные регистры приемника и передатчика располагаются по одному адресу пространства ввода/вывода и обозначаются как регистр данных UDR (Universal Data Register) (UDR*n*). В этом регистре хранятся младшие 8 разрядов принимаемых и передаваемых данных. При чтении выполняется обращение к буферному регистру UDR приемника, при записи — к буферному регистру передатчика. Размещение регистров данных UDR для различных моделей микроконтроллеров приведено в **Табл. 2.123**.

Таблица 2.123. **Размещение регистров данных модулей USART/UART**

Регистр	Адрес	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega32x	ATmega323x	ATmega64x	ATmega128x	Описание
UDR	$0C ($2C)	♦	♦	♦	—	—	♦	♦	♦	—	—	Регистр данных USART/UART
UDR0	$0C ($2C)	—	—	—	♦	♦	—	—	—	♦	♦	Регистр данных USART0/UART0
UDR1	$03 ($23)	—	—	—	♦	♦	—	—	—	—	—	Регистр данных USART1/UART1
	($9C)	—	—	—	—	—	—	—	—	♦	♦	

В модулях USART буфер приемника является двухуровневым (FIFO-буфер), изменение состояния которого происходит при любом обращении к регистру UDR. В связи с этим не следует использовать регистр UDR в качестве операндов команд типа «чтение/модификация/запись» (SBI и CBI). Кроме того, следует быть очень аккуратными при использовании команд проверки SBIC и SBIS, поскольку они также изменяют состояние буфера приемника.

Для управления модулями UART используются два регистра: UCSRA (UCSR*n*A) и UCSRB (UCSR*n*B). А для управления модулями USART используются уже три регистра: UCSRA (UCSR*n*A) , UCSRB (UCSR*n*B) и UCSRC (UCSR*n*C) . Адреса этих регистров указаны в **Табл. 2.124**.

Таблица 2.124. **Регистры управления и состояния модулей USART/UART**

Регистр	Адрес	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega32x	ATmega323x	ATmega64x	ATmega128x	Описание
UCSRA	$0B ($2B)	♦	♦	♦	—	—	♦	♦	♦	—	—	Регистр управления A
UCSRB	$0A ($2A)	♦	♦	♦	—	—	♦	♦	♦	—	—	Регистр управления B
UCSRC	$20 ($40)	♦	♦	♦	—	—	—	♦	♦	—	—	Регистр управления C
UCSR0A	$0B ($2B)	—	—	—	♦	♦	—	—	—	♦	♦	Регистр управления A USART0 (UART0)
UCSR0B	$0A ($2A)	—	—	—	♦	♦	—	—	—	♦	♦	Регистр управления B USART0 (UART0)
UCSR0C	$20 ($40)	—	—	—	—	♦	—	—	—	—	—	Регистр управления C USART0 (UART0)
	($95)	—	—	—	—	—	—	—	—	♦	♦	
UCSR1A	$02 ($22)	—	—	—	♦	♦	—	—	—	—	—	Регистр управления A USART1 (UART1)
	($9B)	—	—	—	—	—	—	—	—	♦	♦	
UCSR1B	$01 ($21)	—	—	—	♦	♦	—	—	—	—	—	Регистр управления B USART1 (UART1)
	($9A)	—	—	—	—	—	—	—	—	♦	♦	
UCSR1C	$3C ($5C)	—	—	—	—	♦	—	—	—	—	—	Регистр управления C USART1 (UART1)
	($9D)	—	—	—	—	—	—	—	—	♦	♦	

Формат регистров UCSRA (UCSR*n*A), UCSRB (UCSR*n*B) и UCSRC (UCSR*n*C) приведен на **Рис. 2.97...2.99**, а значение разрядов этих регистров описано в **Табл. 2.125...2.127** соответственно.

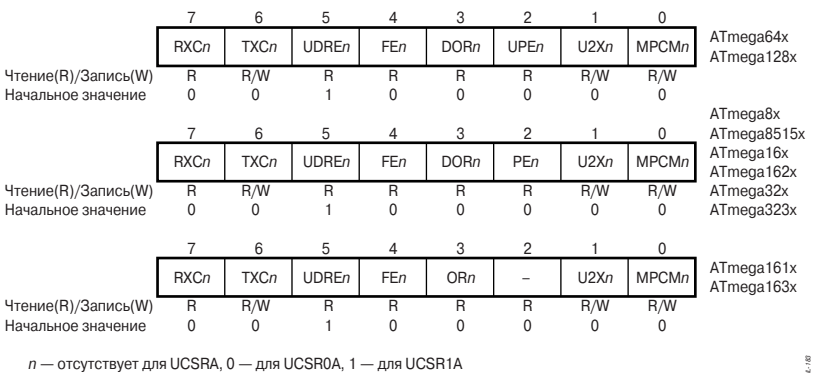


Рис. 2.97. Формат регистров UCSRA, UCSR0A и UCSR1A

Таблица 2.125. **Разряды регистров UCSRA, UCSR0A и UCSR1A**

Разряд	Название	Описание	Модели
7	RXC (RXC*n*)	**Флаг завершения приема.** Флаг устанавливается в «1» при наличии непрочитанных данных в буфере приемника (регистр данных UDR). Сбрасывается флаг аппаратно после опустошения буфера (в UART — после прочтения регистра данных). Если разряд RXCIE (RXCI*n*) регистра UCSRB (UCSR*n*B) установлен, то при установке флага генерируется запрос на прерывание «прием завершен»	Все модели
6	TXC (TXC*n*)	**Флаг завершения передачи.** Флаг устанавливается в «1» после передачи всех разрядов посылки из сдвигового регистра передатчика, при условии, что в регистр данных UDR не было загружено нового значения. Если разряд TXCIE регистра UCSRB (UCSR*n*B) установлен, то при установке флага генерируется прерывание «передача завершена». Флаг сбрасывается аппаратно при выполнении подпрограммы обработки прерывания или программно, записью в него лог. 1	
5	UDRE (UDRE*n*)	**Флаг опустошения регистра данных.** Данный флаг устанавливается в «1» при пустом буфере передатчика (после пересылки байта из регистра данных UDR в сдвиговый регистр передатчика). Установленный флаг означает, что в регистр данных можно загружать новое значение. Если разряд UDRIE регистра UCR (UCSRB) установлен, генерируется запрос на прерывание «регистр данных пуст». Флаг сбрасывается аппаратно, при записи в регистр данных	
4	FE (FE*n*)	**Флаг ошибки кадрирования.** Флаг устанавливается в «1» при обнаружении ошибки кадрирования, т. е. если первый стоп-бит принятой посылки равен «0». Флаг сбрасывается при приеме стоп-бита, равного «1»	
3	OR (OR*n*)	**Флаг переполнения.** В USART флаг устанавливается в «1», если в момент обнаружения нового старт-бита в сдвиговом регистре приемника находится последнее принятое слово, а буфер приемника полон (два значения). В UART флаг устанавливается в «1», если новый кадр будет помещен в сдвиговый регистр приемника до того, как из регистра данных будет считано предыдущее слово. Флаг сбрасывается при пересылке принятых данных из сдвигового регистра приемника в буфер	ATmega161x ATmega163x ATmega8515x*
	DOR (DOR*n*)		Остальные модели

Продолжение таблицы 2.125

Разряд	Название	Описание	Модели
2	PE (PE*n*)	**Флаг ошибки контроля четности.** Флаг устанавливается в «1», если в данных, находящихся в буфере приемника, выявлена ошибка контроля четности. При отключенном контроле четности этот разряд постоянно сброшен в «0»	ATmega8x ATmega8515x ATmega16x ATmega162x ATmega32x ATmega323x
	UPE (UPE*n*)		ATmega64x ATmega128x
	—	Зарезервировано, читается как «0»	ATmega161x ATmega163x ATmega8515x*
1	U2X (U2X*n*)	**Удвоение скорости обмена.** Если этот разряд установлен в «1», коэффициент деления предделителя контроллера скорости передачи уменьшается с 16 до 8, удваивая тем самым скорость асинхронного обмена по последовательному каналу. В USART разряд U2X (U2X*n*) используется только при асинхронном режиме работы. В синхронном режиме он должен быть сброшен	Все модели
0	MPCM (MPCM*n*)	**Режим мультипроцессорного обмена.** Разряд MPCM используется в режиме мультипроцессорного обмена. Если он установлен в «1», ведомый микроконтроллер ожидает приема кадра, содержащего адрес. Кадры, не содержащие адреса устройства, игнорируются	
* В режиме совместимости с AT90S4414/8515.			

Примечание: *n* отсутствует для UCSRA, 0 для UCSRA0 и 1 для UCSRA1.

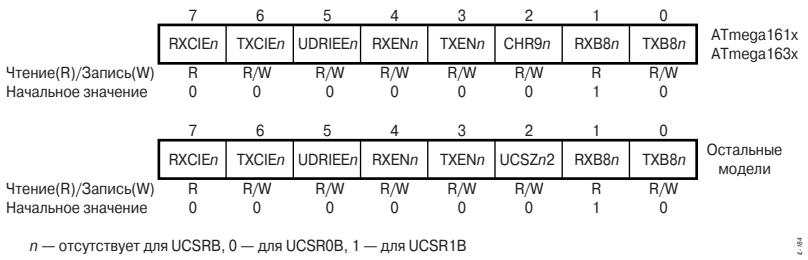


Рис. 2.98. Формат регистров UCSRB, UCSR0B и UCSR1B

Таблица 2.126. **Разряды регистров UCSRB, UCSR0B и UCSR1B**

Разряд	Название	Описание	Модели
7	RXCIE (RXCIE*n*)	**Разрешение прерывания по завершению приема.** Если данный разряд установлен в «1», то при установке флага RXC (RXC*n*) регистра UCSRA (UCSR*n*A) генерируется прерывание «прием завершен» (если флаг I регистра SREG установлен в «1»)	Все модели
6	TXCIE (TXCIE*n*)	**Разрешение прерывания по завершению передачи.** Если данный разряд установлен в «1», то при установке флага TXC (TXC*n*) регистра UCSRA (UCSR*n*A) генерируется прерывание «передача завершена» (если флаг I регистра SREG установлен в «1»)	
5	UDRIE (UDRIE*n*)	**Разрешение прерывания при очистке регистра данных UART.** Если данный разряд установлен в «1», то при установке флага UDRE в регистра UCSRA (UCSR*n*A) генерируется прерывание «регистр данных пуст» (если флаг I регистра SREG установлен в «1»)	
4	RXEN (RXEN*n*)	**Разрешение приема.** При установке этого разряда в «1» разрешается работа приемника USART/UART и переопределяется функционирование вывода RXD (RXD*n*). При сбросе разряда RXEN (RXEN*n*) работа приемника запрещается, а его буфер сбрасывается. Значения флагов TXC(TXC*n*), DOR/OR (DOR*n*/OR*n*) и FE (FE*n*) при этом становятся недействительными	
3	TXEN (TXEN*n*)	**Разрешение передачи.** При установке этого разряда в «1» разрешается работа передатчика UART и переопределяется функционирование вывода TXD (TXD*n*). Если разряд сбрасывается в «0» во время передачи, выключение передатчика произойдет только после завершения передачи данных, находящихся в сдвиговом регистре и буфере передатчика	
2	CHR9 (CHR9*n*)	**Формат посылок.** Этот разряд используется для задания размера слов данных, передаваемых по последовательному каналу. В модулях USART он используется совместно с разрядами UCSZ1:0 (UCSZ*n*1:0) регистра UCSRC (UCSR*n*C). В модулях UART, если разряд CHR9 (CHR9*n*) установлен в «1», осуществляется передача и прием 9-разрядных данных, если сброшен — 8-разрядных	ATmega161x ATmega163x ATmega8515x*
	UCSZ2 (UCSZ*n*2)		Остальные модели
1	RXB8 (RXB8*n*)	**8-й разряд принимаемых данных.** При использовании 9-разрядных слов данных этот разряд содержит значение старшего разряда принятого слова. В случае USART содержимое этого разряда должно быть считано до прочтения регистра данных UDR	Все модели
0	TXB8 (TXB8*n*)	**8-й разряд передаваемых данных.** При использовании 9-разрядных слов данных, содержимое этого разряда является старшим разрядом передаваемого слова. Требуемое значение должно быть занесено в этот разряд до загрузки байта данных в регистр UDR	
* В режиме совместимости с AT90S4414/8515.			

Примечание: *n* = 0 или 1.

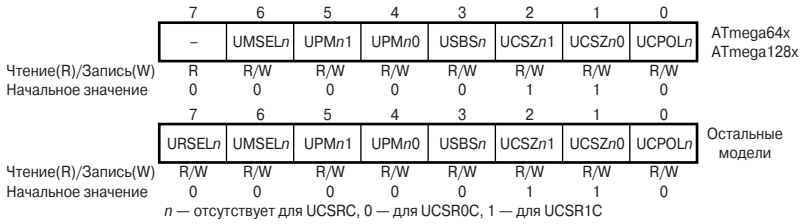


Рис. 2.99. Формат регистров UCSRC, UCSR0C и UCSR1C

Таблица 2.127. **Разряды регистров UCSRC, UCSR0C и UCSR1C**

Разряд	Название	Описание	Модели
7	—	Зарезервировано, читается как «0».	ATmega64x ATmega128x
	URSEL (URSEL*n*)	**Выбор регистра.** Этот разряд определяет, в какой из регистров модуля производится запись. Если разряд установлен в «1», обращение производится к регистру UCSRC (UCSR*n*C). Если же разряд сброшен в «0», обращение производится к регистру UBRRH (UBRR*n*H). Подробнее — см. следующий подраздел	Остальные модели
6	UMSEL (UMSEL*n*)	**Режим работы USART.** Если разряд сброшен в «0», модуль USART работает в асинхронном режиме. Если разряд установлен в «1», то модуль USART работает в синхронном режиме	Все модели
5	UPM1 (UPM*n*1)	**Режим работы схемы контроля и формирования четности.** Эти разряды определяют функционирование схем контроля и формирования четности (см. подраздел 16.2.2)	
4	UPM0 (UPM*n*0)		
3	USBS (USBS*n*)	**Количество стоп-битов.** Этот разряд определяет количество стоп-битов, посылаемых передатчиком. Если разряд сброшен в «0», передатчик посылает 1 стоп-бит, если установлен в «1», то 2 стоп-бита. Для приемника содержимое этого разряда безразлично	
2	UCSZ1 (UCSZ*n*1)	**Формат посылок.** Совместно с разрядом UCSZ2 (UCSZ*n*2) эти разряды определяют количество разрядов данных в посылках (размер слова)	
1	UCSZ0 (UCSZ*n*0)		

Продолжение таблицы 2.127

Разряд	Название	Описание			Модели
0	UCPOL (UCPOL*n*)	**Полярность тактового сигнала.** Значение этого разряда определяет момент выдачи и считывания данных на выводах модуля. Разряд используется только при работе в синхронном режиме. При работе в асинхронном режиме он должен быть сброшен в «0»			Все модели
		UCPOL (UCPOL*n*)	Выдача данных на вывод TXD (TXD*n*)	Считывание данных с вывода RXD (RXD*n*)	
		0	Спадающий фронт XCK (XCK*n*)	Нарастающий фронт XCK (XCK*n*)	
		1	Нарастающий фронт XCK (XCK*n*)	Спадающий фронт XCK (XCK*n*)	

Примечание: *n* = 0 или 1.

16.2.1. Скорость приема/передачи

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

Регистр UBRR является 12-разрядным и физически размещается в двух регистрах ввода/вывода. Адреса и названия этих регистров для различных моделей микроконтроллеров приведены в **Табл. 2.128**.

Таблица 2.128. **Размещение регистров контроллера скорости передачи**

Модель	Регистры	Адрес
ATmega8x*	UBRRH:UBRRL	$20 ($40):$09 ($29)
ATmega8515x*	UBRRH:UBRRL	$20 ($40):$09 ($29)
ATmega16x*	UBRRH:UBRRL	$20 ($40):$09 ($29)
ATmega161x	UBRRHI[3:0]:UBRR0	$20 ($40):$09 ($29)
	UBRRHI[7:4]:UBRR1	$20 ($40):$00 ($20)
ATmega162x*	UBRR0H:UBRR0L	$20 ($40):$09 ($29)
	UBRR1H:UBRR1L	$3C ($5C):$00 ($20)
ATmega163x	UBRRHI:UBRR	$20 ($40):$09 ($29)
ATmega32x*	UBRRH:UBRRL	$20 ($40):$09 ($29)
ATmega323x*	UBRRH:UBRRL	$20 ($40):$09 ($29)

Продолжение таблицы 2.128

Модель	Регистры	Адрес
ATmega64x/128x	UBRR0H:UBRR0L	($90):$09 ($29)
	UBRR1H:UBRR1L	($98):($99)
* В этих моделях регистр UBRRH (UBRR*n*H) размещается по тому же адресу, что и регистр UCSRC (UCSR*n*C).		

Следует иметь в виду, что в моделях ATmega8x, ATmega8515x, ATmega16x, ATmega162x, ATmega32x и ATmega323x регистр UBRRH (UBRR*n*H) размещается по тому же адресу, что и регистр управления UCSRC (UCSR*n*C). Поэтому при обращении по этим адресам необходимо выполнить ряд дополнительных действий для выбора конкретного регистра.

При записи регистр определяется состоянием старшего разряда записываемого значения URSEL (URSEL*n*). Если этот разряд сброшен в «0», изменяется содержимое регистра UBRRH (UBRR*n*H). Если же старший разряд значения установлен в «1», изменяется содержимое регистра управления UCSRC (UCSR*n*C). Приведенные ниже фрагменты программ иллюстрируют сказанное:

Пример на ассемблере

```
...
;Записать 2 в UBRRH
ldi r16,0x02
out UBRRH,r16
...
;Установить разряды USBS и UCSZ1 регистра UCSRC
ldi r16(1<<URSEL)|(1<<USBS)|(1<<UCSZ1)
out UCSRC,r16
...
```

Пример на C

```
...
/* Записать 2 в UBRRH */
UBRRH = 0x02
...
/* Установить разряды USBS и UCSZ1 регистра UCSRC */
UCSRC = (1<<URSEL)|(1<<USBS)|(1<<UCSZ1);
...
```

Для выбора регистра при чтении используется временная последовательность. При первом обращении по указанным адресам возвращается

значение регистра UBRRH (UBRRH*n*). При повторном обращении по этим адресам в следующем машинном цикле возвращается значение регистра UCSRC (UCSR*n*C), как показано в приведенных ниже примерах. Прерывания при выполнении этой последовательности команд должны быть запрещены.

Пример на ассемблере

```
USART_ReadUCSRC:
                              ;Прочитать регистр UCSRC
in r16,UBRRH
in r16,UCSRC
ret                           ;Значение регистра UCSRC возвращается
                              в регистре r16
```

Пример на C

```
unsigned char USART_ReadUCSRC(void)
{
unsigned char ucsrc;
ucsrc = UBRRH;
ucsrc = UCSRC;
return ucsrc;
}
```

При работе в асинхронном режиме скорость обмена определяется не только содержимым регистра UBRR, но и состоянием разряда U2X (U2X*n*) регистра UCSRA (UCSR*n*A). Если этот разряд установлен в «1», коэффициент деления предделителя уменьшается в два раза, а скорость обмена соответственно удваивается. При работе в синхронном режиме этот разряд должен быть сброшен.

Итак, скорость обмена определяется следующими формулами, где BAUD — скорость передачи в бодах, $f_{СК}$ — тактовая частота микроконтроллера, UBRR — содержимое регистра контроллера скорости передачи (0...4095):

- асинхронный режим (обычный, U2X*n* = «0»)
  $BAUD = f_{СК}/16(UBRR + 1)$;
- асинхронный режим (ускоренный, U2X*n* = «1»)
  $BAUD = f_{СК}/8(UBRR + 1)$;
- синхронный режим ведущего
  $BAUD = f_{СК}/2(UBRR + 1)$.

В качестве примера в **Табл. 2.129** приведены значения регистра UBRR, позволяющие получить стандартные для асинхронного режима скорости передачи при использовании различных резонаторов, а также величины ошибок получаемых значений относительно стандартных.

Таблица 2.129. **Пример установок регистра UBRR**

Скорость [бод]	1 МГц				1.8432 МГц				2 МГц			
	U2X = «0»		U2X = «1»		U2X = «0»		U2X = «1»		U2X = «0»		U2X = «1»	
	UBRR	*[%]	UBRR	*[%]	UBRR	*[%]	UBRR	*[%]	UBRR	*[%]	UBRR	*[%]
2400	25	0.2	51	0.2	47	0.0	95	0.0	51	0.2	103	0.2
4800	12	0.2	25	0.2	23	0.0	47	0.0	25	0.2	51	0.2
9600	6	7.5	12	0.2	11	0.0	23	0.0	12	0.2	25	0.2
14400	3	7.8	8	–3.5	7	0.0	15	0.0	8	–3.5	16	2.1
19200	2	7.8	6	–7.0	5	0.0	11	0.0	6	–7.0	12	0.2
28800	1	7.8	3	8.5	3	0.0	7	0.0	3	8.5	8	–3.5
38400	1	22.9	2	8.5	2	0.0	5	0.0	2	8.5	6	–7.0
57600	0	7.8	1	8.5	1	0.0	3	0.0	1	8.5	3	8.5
76800	—	—	1	–18.6	1	–25.0	2	0.0	1	–18.6	2	8.5
115200	—	—	0	8.5	0	0.0	1	0.0	0	8.5	1	8.5
230400	—	—	—	—	—	—	0	0.0	—	—	—	—
250000	—	—	—	—	—	—	—	—	—	—	0	0.0
Max	62.5 кбод		125 кбод		115.2 кбод		230.4 кбод		125 кбод		250 кбод	

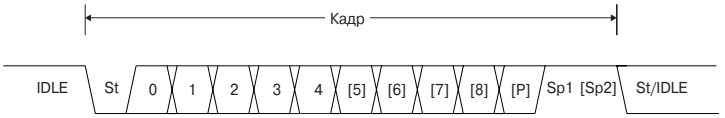
* Отклонение скорости передачи.

При UBRR = 0 ошибка = 0.0%. Рекомендуется использовать значения регистра UBRR, при которых получаемая скорость передачи отличается от требуемого значения меньше чем на 0.5%. Значения, дающие большее отклонение также можно использовать, однако следует иметь в виду, что при этом снижается помехозащищенность линии передачи.

При работе в синхронном режиме в качестве ведомого скорость приема и передачи определяется частотой сигнала, поступающего на вывод XCK (XCK*n*). Частота этого сигнала должна удовлетворять выражению $f_{XCK} < f_{OSC}/4$. Это ограничение связано с тем, что сигнал с вывода XCK (XCK*n*) сначала синхронизируется с тактовой частотой микроконтроллера, а затем проходит через детектор фронтов. Задержка сигнала при прохождении этих узлов равна двум машинным циклам.

16.2.2. Формат кадра

Под кадром в данном случае понимается совокупность одного слова данных и сопутствующей информации (**Рис. 2.100**). Кадр начинается со старт-бита, за которым следует младший разряд слова данных. После старшего разряда слова данных следует один или два стоп-бита. Если включена схема формирования бита четности, он включается между старшим разрядом слова данных и первым стоп-битом.



St — старт-бит, всегда «0»
(0...8) — разряды данных
P — бит четности
Sp1 [Sp2] — стоп-бит, всегда «1»
IDLE — нет обмена по линии RxD или TxD (должна быть «1»)

Рис. 2.100. Формат кадра

Формат кадра определяется несколькими разрядами регистров UCSRB (UCSR*n*B) и UCSRC (UCSR*n*C). Размер слова данных в USART определяется разрядами UCSZ2...UCSZ0 (UCSZ*n*2...UCSZ*n*0) в соответствии с **Табл. 2.130**.

Таблица 2.130. **Определение размера слова данных в модулях USART**

UCSZ2 (UCSZ*n*2)	UCSZ1 (UCSZ*n*1)	UCSZ0 (UCSZ*n*0)	Размер слова данных
0	0	0	5 разрядов
0	0	1	6 разрядов
0	1	0	7 разрядов
0	1	1	8 разрядов
1	0	0	Зарезервировано
1	0	1	Зарезервировано
1	1	0	Зарезервировано
1	1	1	9 разрядов

Примечание: *n* = 1 или 2.

В модулях UART слово данных может быть только 8- или 9-разрядным, что определяется состоянием флага CHR9 (CHR9*n*) регистра UCSRB (UCSR*n*B). Если этот флаг сброшен в «0», размер слова равен 8 разрядам, если установлен в «1» — 9 разрядам.

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

Разряды UPM1:UPM0 (UPM*n*1:UPM*n*0) регистра UCSRX (UCSR*n*C) определяют функционирование схемы контроля четности модулей USART согласно **Табл. 2.131**.

Таблица 2.131. **Управление контролем четности**

UPM1 (UPM*n*1)	UPM0 (UPM*n*0)	Режим работы
0	0	Выключен
0	1	Зарезервировано
1	0	Включен, проверка на четность (even parity)
1	1	Включен, проверка на нечетность (odd parity)

Примечание: *n* = 1 или 2.

Значение бита четности получается путем выполнения операции «Исключающее ИЛИ» над всеми разрядами передаваемого слова данных. Если используется проверка на нечетность (odd parity), полученный результат инвертируется:

Если контроль четности включен, бит четности, как уже было сказано, вставляется передатчиком между старшим разрядом передаваемых данных и первым стоп-битом.

16.2.3. Передача данных

Работа передатчика разрешается установкой в «1» разряда TXEN (TXEN*n*) регистра UCSRB (UCSR*n*B). При установке разряда вывод TXD (TXD*n*) подключается к передатчику USART/UART и начинает функционировать как выход независимо от установок регистров управления портом. Если используется синхронный режим работы (в USART), переопределяется также функционирование вывода XCK (XCK*n*).

Передача инициируется записью передаваемых данных в буферный регистр передатчика — регистр данных UDR. После этого данные пересылаются из регистра UDR в сдвиговый регистр передатчика. Одновременно, если используются 9-разрядные данные, значение разряда TXB8 (TXB8*n*) регистра UCSRB (UCSR*n*B) копируется в 9-й разряд сдвигового регистра. При этом возможны два варианта:

- запись в регистр UDR осуществляется в тот момент, когда передатчик находится в состоянии ожидания (предыдущие данные уже переданы). В этом случае данные пересылаются в сдвиговый регистр сразу же после записи в регистр UDR;

- запись в регистр UDR осуществляется во время передачи. В этом случае данные пересылаются в сдвиговый регистр после передачи последнего стоп-бита текущего кадра.

Очевидно, что 9-й разряд данных должен быть загружен в разряд TXB8 (TXB8*n*) до записи младшего байта слова в регистр данных.

После пересылки слова данных в сдвиговый регистр, флаг UDRE (UDRE*n*) регистра UCSRA (UCSR*n*A) устанавливается в «1», что означает готовность передатчика к получению нового слова данных. В этом состоянии флаг остается до следующей записи в буфер. Одновременно с пересылкой в регистре формируется служебная информация — старт-бит, возможный бит четности (только в USART), а также один или два стоп-бита.

После загрузки сдвигового регистра его содержимое начинает сдвигаться вправо и поступать на вывод TXD (TXD*n*) в порядке, рассмотренном в подразделе 16.2.2. Скорость сдвига определяется настройками контроллера тактовых сигналов. При работе в синхронном режиме (только USART) изменение состояния вывода TXD (TXD*n*) происходит по одному из фронтов сигнала XCK (XCK*n*). Если разряд UCPOL (UCPOL*n*) регистра UCSRC (UCSR*n*C) сброшен в «0», изменение состояния вывода происходит по нарастающему фронту сигнала XCK (XCK*n*), если же установлен в «1» — по спадающему фронту сигнала, как показано на **Рис. 2.101**.

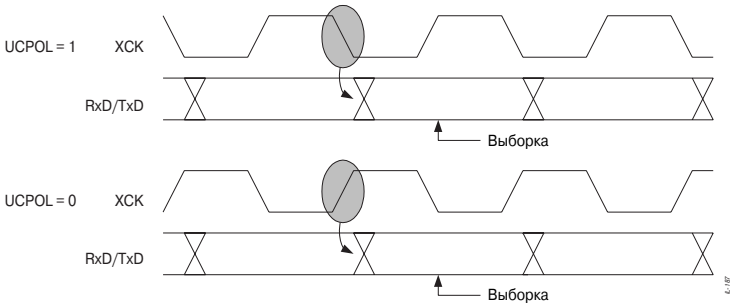


Рис. 2.101. Временные диаграммы для синхронного режима работы USART

Если во время передачи в регистр UDR было записано новое слово данных, то после передачи последнего стоп-бита оно пересылается в сдвиговый регистр. Если же к моменту окончания передачи кадра такой записи выполнено не было, устанавливается флаг прерывания «Передача завершена» TXC (TXC*n*) регистра UCSRA (UCSR*n*A). Сброс флага осуществляется аппаратно при входе в подпрограмму обработки соответствующего прерывания или программно, записью в этот разряд лог. 1.

Выключение передатчика осуществляется сбросом разряда TXEN (TXEN*n*) регистра UCSRB (UCSR*n*B). Если в момент выполнения этой

команды осуществлялась передача, сброс разряда произойдет только после завершения текущей и отложенной передач, т. е. после очистки сдвигового и буферного регистров передатчика. При выключенном передатчике вывод TXD (TXD*n*) может использоваться как контакт ввода/вывода общего назначения.

Ниже приведен простейший пример подпрограммы передачи по интерфейсу USART/UART. Эта подпрограмма ждет очистки буфера передатчика, а затем загружает в него новое значение.

Пример на ассемблере

```
USART_Transmit:
;Ждать очистки буфера передатчика
sbis UCSRA,UDRE
rjmp USART_Transmit
;Скопировать 9-й разряд данных из r17 в TXB8
cbi UCSRB,TXB8
sbrc r17,0
sbi UCSRB,TXB8
;Загрузить младший байт данных в буфер, начать передачу
out UDR,r16
ret
```

Пример на C

```
void USART_Transmit(unsigned int data)
{
/* ждать очистки буфера передатчика */
while ( !( UCSRA & (1<<UDRE)))
;
/* скопировать 9-й разряд данных из r17 в TXB8 */
UCSRB &= ~(1<<TXB8);
if (data & 0x100)
UCSRB |= (1<<TXB8);
/* загрузить младший байт данных в буфер, начать передачу */
UDR = data
}
```

16.2.4. Прием данных

Работа приемника разрешается установкой разряда RXEN (RXEN*n*) регистра UCSRB (UCSR*n*B). При установке разряда вывод RXD (RXD*n*) подключается к приемнику USART/UART и начинает функционировать как

вход независимо от установок регистров управления портом. Если используется синхронный режим работы (в USART), переопределяется также функционирование вывода XCK (XCK*n*).

Прием данных начинается сразу же после обнаружения приемником корректного старт-бита. Каждый разряд содержимого кадра затем считывается с частотой, определяемой установками контроллера скорости передачи или тактовым сигналом XCK (XCK*n*). Считанные разряды данных последовательно помещаются в сдвиговый регистр приемника до обнаружения первого стоп-бита кадра. После этого содержимое сдвигового регистра пересылается в буфер приемника, из которого принятое значение может быть получено путем чтения регистра данных модуля. При использовании 9-разрядных слов данных значение старшего разряда может быть определено по состоянию флага RX8 (RX8*n*) регистра UCSRB (UCSR*n*B). Причем в модулях USART содержимое старшего разряда данных должно быть считано до обращения к регистру данных. Это связано с тем, что флаг RX8 (RX8*n*) отображает значение старшего разряда слова данных кадра, находящегося на верхнем уровне буфера приемника, состояние которого при чтении регистра данных изменится.

Если во время приема кадра была включена схема контроля четности (только USART), она вычисляет бит четности для всех разрядов принятого слова данных и сравнивает его с принятым битом четности. Результат проверки запоминается в буфере приемника вместе с принятым словом данных и стоп-битами. Наличие или отсутствие ошибки контроля четности может быть затем определено по состоянию флага UPE (UPE*n*). Этот флаг устанавливается в «1», если следующее слово, которое может быть прочитано из буфера, имеет ошибку контроля четности. При выключенном контроле четности флаг UPE (UPE*n*) всегда читается как «0».

Блок приемника модулей USART/UART имеет еще два флага, показывающих состояние обмена: флаг ошибки кадрирования FE (FE*n*) и флаг переполнения DOR (DOR*n*)/OR (OR*n*). Флаг FE (FE*n*) устанавливается в «1», если значение первого стоп-бита принятого кадра не соответствует требуемому, т. е. равно «0».

Флаги DOR (DOR*n*) в USART и OR (OR*n*) в UART индицируют потерю данных из-за переполнения буфера приемника. В UART флаг устанавливается в «1», если к моменту окончания приема кадра (заполнения сдвигового регистра приемника) данные предыдущего кадра не были считаны из регистра данных. В USART флаг устанавливается в «1» в случае приема старт-бита нового кадра при заполненных буфере и сдвиговом регистре приемника. Установленный флаг DOR (DOR*n*)/OR (OR*n*) означает, что между прошлым байтом, считанным из регистра UDR, и байтом, считанным в данный момент, произошла потеря одного или нескольких кадров.

Следует иметь в виду, что обработка описанных флагов в модулях UART и USART несколько отличается. В модулях UART флаг ошибки кадрирования FE (FE*n*) должен быть прочитан перед обращением к регистру данных, а флаг переполнения OR (OR*n*) — после обращения к этому регистру.

В модулях USART все флаги ошибок буферизуются вместе со словом данных, т. е. соответствующие разряды регистра UCSRA (UCSR*n*A) относятся к кадру, слово данных которого будет прочитано при следующем обращении к регистру данных UDR (UDR*n*). Поэтому состояние этих флагов должно быть считано перед обращением к регистру данных. Кроме того, для совместимости с будущими устройствами рекомендуется при записи в регистр UCSRA (UCSR*n*A) сбрасывать соответствующие этим флагам разряды записываемого значения в «0».

Для индикации состояния приемника в модулях USART/UART используется флаг прерывания «Прием завершен» RXC (RXC*n*) регистра UCSRA (UCSR*n*A). Этот флаг устанавливается в «1» при наличии в буфере приемника непрочитанных данных. В модулях UART этот флаг сбрасывается после прочтения регистра данных, а в модулях USART — при опустошении буфера (после считывания всех находящихся в нем данных).

Выключение приемника осуществляется сбросом разряда RXEN (RXEN*n*) регистра UCSRB (UCSR*n*B). В отличие от передатчика приемник выключается сразу же после сброса разряда, т. е. кадр, принимаемый в этот момент, теряется. В модулях USART, кроме того, при выключении приемника очищается его буфер, т. е. теряются также все непрочитанные данные. При выключенном приемнике вывод RXD (RXD*n*) может использоваться как контакт ввода/вывода общего назначения.

Пример подпрограммы приема по интерфейсу USART приведен ниже. Как и в предыдущем примере, здесь используется опрос флага прерывания.

Пример на ассемблере

```
USART_Receive:
;Ждать загрузки данных в буфер приемника
sbis UCSRA,RXC
rjmp USART_Receive
;Прочитать 9-й разряд данных и флаги состояния
in r18,UCSRA
in r17,UCSRB
;Прочитать младший байт данных
in r16,UDR
;В случае ошибки вернуть -1
andi r18,(1<<FE)|(1<<DOR)|(1<<UPE)
breq no_error
```

```
USART_Receive:
ldi r17,HIGH(-1)
ldi r18,LOW(-1)
no_error:
;Выделить 9-й разряд данных
lsr r17
andi r17,0x01
ret
```

Пример на С

```
unsigned int USART_Receive(void)
{
unsigned char status, resh, resl
/* Ждать заполнения буфера приемника */
while(!(UCSRA & (1<<RXC)))
;
/* Прочитать 9-й разряд данных и флаги состояния */
status = UCSRA;
resh = UCSRB;
/* Прочитать младший байт данных */
resl = UDR;
/* В случае ошибки вернуть -1*/
if (status & (1<<FE)|(1<<DOR)|(1<<UPE))
return -1;
/* Выделить 9-й разряд данных */
resh = (resh>>1) & 0x01;
return ((resh<<8) | resl);
}
```

Собственно прием всех разрядов кадра осуществляется по-разному, в зависимости от режима работы модуля. При работе модуля USART в синхронном режиме состояние вывода RXD (RXD*n*) считывается по одному из фронтов сигнала XCK (XCK*n*). Если разряд UCPOL (UCPOL*n*) регистра UCSRC (UCSR*n*C) сброшен в «0», считывание состояния вывода происходит по спадающему фронту сигнала XCK (XCK*n*), если же установлен в «1» — по нарастающему фронту сигнала. Другими словами, считывание данных с вывода RXD (RXD*n*) и их выдача на вывод TXD (TXD*n*) происходят по противоположным фронтам (**Рис. 2.101**).

Для обеспечения приема в асинхронном режиме работы используются схемы восстановления тактового сигнала и данных. Схема восстановления

тактового сигнала предназначена для синхронизации внутреннего тактового сигнала, формируемого контроллером скорости передачи, и кадров, поступающих на вывод RXD (RXD*n*) микроконтроллера. Схема восстановления данных осуществляет считывание и фильтрацию каждого разряда принимаемого кадра.

Схема восстановления тактового сигнала осуществляет опрос входа приемника с целью определения старт-бита кадра. Частота опроса зависит от состояния разряда U2X (U2X*n*) регистра UCSRA (UCSR*n*A). В обычном режиме (при U2X*n* = «0») частота опроса в 16 раз превышает скорость передачи данных, а в ускоренном режиме (при U2X = «1») — в 8 раз.

Обнаружение изменения сигнала на выводе RXD (RXD*n*) с лог. 1 (режим ожидания) на лог. 0 интерпретируется как возможное появление переднего фронта старт-бита. После этого в нормальном режиме проверяется значение 8-й, 9-й и 10-й выборок входного сигнала, а в ускоренном режиме — 4-й, 5-й и 6-й выборок (**Рис. 2.102, а**). Если значение хотя бы двух выборок из указанных равно лог. 1, старт-бит считается ложным (помеха), а приемник переходит к ожиданию следующего изменения входного сигнала с лог. 1 на лог. 0. В противном случае считается, что обнаружен старт-бит новой последовательности, с которым синхронизируется внутренний тактовый сигнал приемника. После этого начинает работать схема восстановления данных.

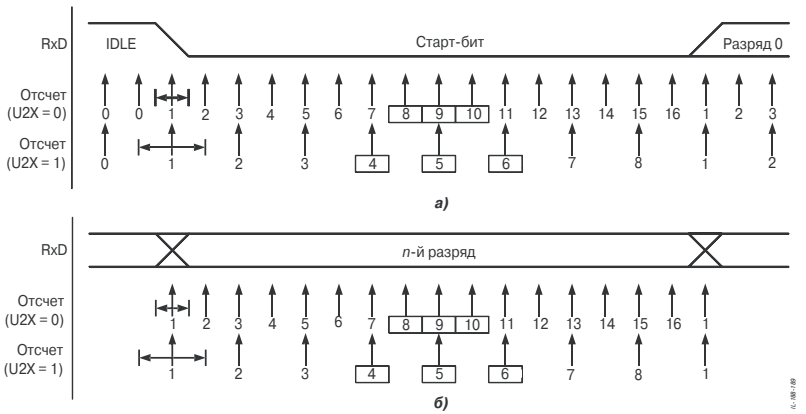


Рис. 2.102. Распознавание разрядов кадра:
а — старт-бит; *б* — остальные разряды

Решение о значении принятого разряда принимается также по результатам 8-й, 9-й и 10-й (4-й, 5-й и 6-й) выборок входного сигнала

(**Рис. 2.102, б**). Состоянием разряда считается логическое значение, которое было получено по меньшей мере в двух из трех выборок. Процесс распознавания повторяется для всех разрядов принимаемого кадра, включая первый стоп-бит.

Из сказанного следует, что старт-бит нового кадра может передаваться сразу же после последней выборки, используемой для определения значения разряда. В обычном режиме работы формирование старт-бита может начаться в момент A, а в ускоренном режиме — в момент B (**Рис. 2.103**). Момент C, обозначенный на рисунке, определяет максимальную длительность стоп-бита.

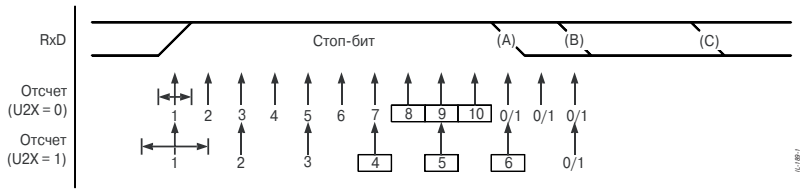


Рис. 2.103. Распознавание стоп-бита и последующего старт-бита

16.3. Мультипроцессорный режим работы

Режим мультипроцессорного обмена позволяет осуществлять связь между несколькими ведомыми микроконтроллерами и одним ведущим. В этом режиме каждый ведомый микроконтроллер имеет свой уникальный адрес. Если ведущий микроконтроллер хочет что-либо передать, он посылает адресный байт, определяющий, к какому из микроконтроллеров он собирается обратиться. Если какой-либо из ведомых микроконтроллеров распознал свой адрес, он переходит в режим приема данных и соответственно принимает последующие байты как данные. Остальные ведомые микроконтроллеры игнорируют принимаемые байты до посылки ведущим нового адресного байта. Включение режима фильтрации принимаемых кадров, не содержащих адреса, осуществляется установкой в «1» разряда MPCM (MPCM*n*) регистра UCSRA (UCSR*n*A).

В микроконтроллере, выполняющем роль ведущего, должен быть установлен режим передачи 9-разрядных данных. При передаче адресного байта старший разряд должен устанавливаться в «1», а при передаче байтов данных он должен сбрасываться в «0».

В ведомых микроконтроллерах механизм приема зависит от режима работы приемника. Если приемник настроен на прием 5...8-разрядных данных, то идентификация содержимого кадра (адрес/данные) осуществляется по 1-му стоп-биту. При приеме 9-разрядных данных идентификация содержимого кадра осуществляется по старшему разряду слова данных.

Если указанные разряды установлены в «1», значит, кадр содержит адрес, если — в «0», кадр содержит данные.

Для осуществления обмена данными в многопроцессорном режиме необходимо действовать следующим образом:

1. Все ведомые микроконтроллеры переключаются в режим мультипроцессорного обмена установкой в «1» разряда MPCM (MPCM*n*) регистра UCSRA (UCSR*n*A).
2. Ведущий микроконтроллер посылает адресный кадр, а все ведомые микроконтроллеры его принимают. Соответственно в каждом из ведомых микроконтроллеров устанавливается флаг RXC (RXC*n*) регистра UCSRA (UCSR*n*A).
3. Каждый из ведомых микроконтроллеров считывает содержимое регистра данных. Микроконтроллер, адрес которого совпал с адресом, посланным ведущим, сбрасывает в «0» разряд MPCM (MPCM*n*).
4. Адресованный микроконтроллер начинает принимать кадры, содержащие данные. Если приемник ведомого микроконтроллера настроен на прием 5...8-разрядных данных, будет генерироваться ошибка кадрирования, поскольку первый стоп-бит будет равен «0» (в USART этого можно избежать, используя два стоп-бита). В остальных ведомых микроконтроллерах разряд MPCM (MPCM*n*) установлен в «1», поэтому кадры данных будут игнорироваться.
5. После приема последнего байта данных адресованный микроконтроллер устанавливает в «1» разряд MPCM (MPCM*n*) и снова ожидает приход кадра с адресом. Процесс повторяется с пункта 2.

Глава 17. **Последовательный периферийный интерфейс SPI**

17.1. Введение

Последовательный периферийный интерфейс SPI (Serial Peripheral Interface), реализованный во всех микроконтроллерах семейства Mega, имеет двоякое назначение. Во-первых, с его помощью может осуществляться обмен данными между микроконтроллером и различными периферийными устройствами, такими, как цифровые потенциометры, ЦАП/АЦП, FLASH-ПЗУ и др. Посредством этого интерфейса также может производиться обмен данными между несколькими микроконтроллерами AVR. Использование интерфейса SPI в качестве высокоскоростного канала связи и рассматривается в данной главе.

Кроме того, через интерфейс SPI может быть осуществлено программирование микроконтроллера (т. н. режим последовательного программирования). Использование интерфейса SPI в этом качестве будет описано в 4-й части книги.

При обмене данными по интерфейсу SPI микроконтроллер AVR может работать как ведущий (режим «Master») либо как ведомый (режим «Slave»). При этом пользователь может задавать скорость передачи (семь программируемых значений) и формат передачи (от младшего разряда к старшему или наоборот).

Дополнительной возможностью подсистемы SPI является «пробуждение» микроконтроллера из режима Idle при поступлении данных.

17.2. Функционирование модуля SPI

Структурная схема модуля SPI приведена на **Рис. 2.104**.

Модуль SPI использует четыре вывода микроконтроллера. Как и для большинства прочих периферийных устройств, эти выводы являются линиями портов ввода/вывода общего назначения (**Табл. 2.132**).

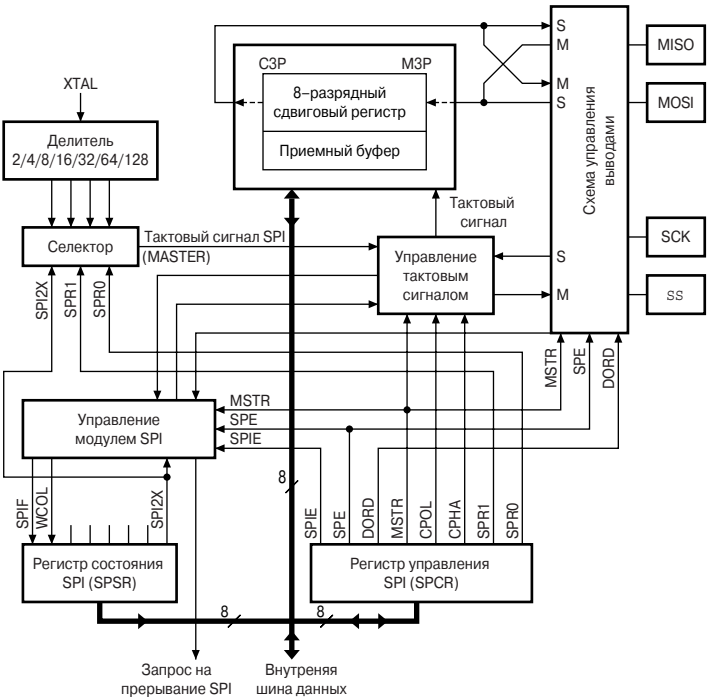


Рис. 2.104. Структурная схема модуля SPI

Таблица 2.132. **Выводы, используемые модулем SPI**

Вывод	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega32x	ATmega323x	ATmega323x	ATmega64x ATmega128x	Назначение
SCK	PB5	PB7	PB7	PB7	PB7	PB7	PB7	PB7	PB7	PB1	Выход (master)/вход (slave) тактового сигнала
MISO	PB4	PB6	PB6	PB6	PB6	PB6	PB6	PB6	PB6	PB3	Вход (master)/выход (slave) данных
MOSI	PB3	PB5	PB5	PB5	PB5	PB5	PB5	PB5	PB5	PB2	Выход (master)/вход (slave) данных
$\overline{SS}$	PB2	PB4	PB4	PB4	PB4	PB4	PB4	PB4	PB4	PB0	Выбор ведомого устройства

При включенном модуле SPI режим работы указанных выводов (направление передачи данных) переопределяется согласно **Табл. 2.133**. Направление передачи данных определяется состоянием соответствующего разряда регистра DDRB.

Таблица 2.133. **Переназначение режима работы выводов модуля SPI**

Вывод	Режим «Master»	Режим «Slave»
MOSI	Определяется пользователем	Вход
MISO	Вход	Определяется пользователем
SCK	Определяется пользователем	Вход
$\overline{SS}$	Определяется пользователем	Вход

Как видно из таблицы, в определенных случаях пользователь должен самостоятельно задать режим работы вывода, используемого модулем SPI, в соответствии с его назначением (см. далее). При этом возможность управления внутренними подтягивающими резисторами выводов, работающих как входы, сохраняется независимо от способа управления их режимом работы.

Для управления модулем SPI предназначен регистр управления SPCR, расположенный по адресу $0D ($2D). Формат этого регистра приведен на **Рис. 2.105**, а краткое описание функций разрядов регистра приведено в **Табл. 2.134**. Подробно использование различных разрядов регистра будет описано далее.

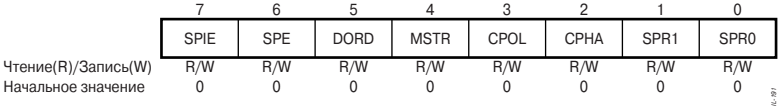


Рис. 2.105. Формат регистра SPCR

Таблица 2.134. **Разряды регистра SPCR**

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

Контроль состояния модуля, а также дополнительное управление скоростью обмена осуществляется с помощью регистра состояния SPSR, распо-

ложенного по адресу $0E ($2E). Разряды с 7-го по 1-й этого регистра доступны только для чтения, а 0-й разряд — как для чтения, так и для записи. Формат этого регистра приведен на **Рис. 2.106**, а назначение его разрядов описано в **Табл. 2.135**.

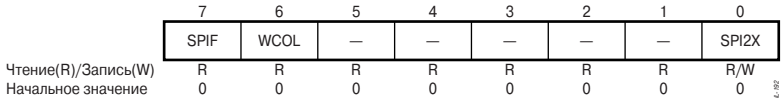


Рис. 2.106. Формат регистра SPSR

Таблица 2.135 **Разряды регистра SPSR**

Разряд	Название	Описание
7	SPIF	**Флаг прерывания от SPI.** Данный флаг устанавливается в «1» по окончании передачи очередного байта. Если флаг SPIE регистра SPCR установлен в «1» и прерывания разрешены, одновременно с установкой флага генерируется прерывание от SPI. Также флаг SPIF устанавливается в «1» при переводе микроконтроллера из режима «Master» в режим «Slave» посредством вывода $\overline{SS}$ (см. 17.3). Флаг сбрасывается аппаратно, либо при старте подпрограммы обработки прерывания, либо после чтения регистра состояния SPI с последующим обращением к регистру данных SPI (SPDR)
6	WCOL	**Флаг конфликта записи.** Данный флаг устанавливается в «1» при попытке записи в регистр данных (SPDR) во время передачи очередного байта. Флаг сбрасывается аппаратно после чтения регистра состояния SPI с последующим обращением к регистру данных SPI
5...1	—	Зарезервированы, читаются как «0»
0	SPI2X	**Удвоение скорости обмена.** При установке этого разряда в «1» и работе микроконтроллера в режиме «Master» частота сигнала SCK удваивается

Передаваемые данные записываются, а принимаемые — считываются из регистра данных SPDR, расположенного по адресу $0F ($2F). Запись в этот регистр инициирует начало передачи, а при его чтении считывается содержимое буфера сдвигового регистра. Другими словами, регистр данных служит буфером между регистровым файлом микроконтроллера и сдвиговым регистром модуля SPI.

Соединение двух микроконтроллеров (ведущий—ведомый) по интерфейсу SPI показано на **Рис. 2.107**. Вывод SCK ведущего микроконтроллера является выходом тактового сигнала, а ведомого микроконтроллера — входом.

Перед выполнением обмена необходимо, прежде всего, разрешить работу модуля SPI. Для этого следует установить в «1» разряд SPE регистра SPCR. Режим работы определяется состоянием разряда MSTR этого регистра: если разряд установлен в «1», микроконтроллер работает в режиме «Master», если сброшен в «0» — в режиме «Slave».

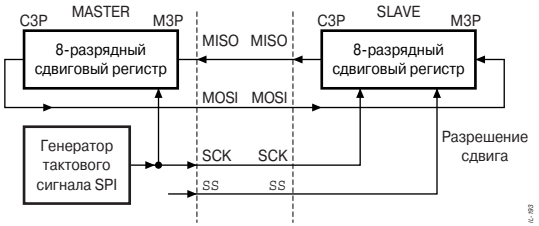


Рис. 2.107. Соединение микроконтроллеров по интерфейсу SPI

Передача данных осуществляется следующим образом. При записи в регистр данных SPI ведущего микроконтроллера запускается генератор тактового сигнала модуля SPI, и данные начинают поразрядно выдаваться на вывод MOSI и соответственно поступать на вывод MOSI ведомого микроконтроллера. Порядок передачи разрядов данных определяется состоянием разряда DORD регистра SPCR. Если разряд установлен в «1», первым передается младший разряд байта, если же сброшен в «0» — старший разряд. После выдачи последнего разряда текущего байта генератор тактового сигнала останавливается с одновременной установкой в «1» флага «Конец передачи» (SPIF). Если прерывания от модуля SPI разрешены (флаг SPIE регистра SPCR установлен в «1»), генерируется запрос на прерывание. После этого ведущий микроконтроллер может начать передачу следующего байта либо, подав на вход $\overline{SS}$ ведомого микроконтроллера напряжение ВЫСОКОГО уровня, перевести последний в состояние ожидания.

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

В модуле реализована одинарная буферизация при передаче и двойная при приеме. Это означает, что готовый для передачи байт данных не может быть записан в регистр данных SPI до окончания предыдущего цикла обмена. При попытке изменить содержимое регистра данных во время передачи устанавливается в «1» флаг WCOL регистра SPSR. Сбрасывается этот флаг после чтения регистра SPSR с последующим обращением к регистру данных SPI.

Соответственно, при приеме данных принятый байт должен быть прочитан из регистра данных SPI до того, как в сдвиговый регистр поступит последний разряд следующего байта. В противном случае первый байт будет потерян.

17.3. Режимы передачи данных

Спецификация интерфейса SPI предусматривает 4 режима передачи данных. Эти режимы различаются соответствием между фазой (момент считывания сигнала) тактового сигнала SCK, его полярностью и передаваемыми данными. Всего существует 4 таких комбинации, определяемых состоянием разрядов CPHA и CPOL регистра SPCR (**Табл. 2.136**).

Таблица 2.136. **Задание режима передачи данных**

Разряд	Описание
CPOL	**Полярность тактового сигнала.** «0» — генерируются импульсы положительной полярности, при отсутствии импульсов на выводе присутствует НИЗКИЙ уровень; «1» — генерируются импульсы отрицательной полярности, при отсутствии импульсов на выводе присутствует ВЫСОКИЙ уровень
CPHA	**Фаза тактового сигнала.** «0» — обработка данных производится по переднему фронту импульсов сигнала SCK (для CPOL = «0» — по нарастающему фронту, а для CPOL = «1» — по спадающему фронту); «1» — обработка данных производится по заднему фронту импульсов сигнала SCK (для CPOL = «0» — по спадающему фронту, а для CPOL = «1» — по нарастающему фронту)

Соответствующие этим режимам форматы обмена данными через SPI приведены на **Рис. 2.108** и **Рис. 2.109** (передача ведется от старшего разряда к младшему).

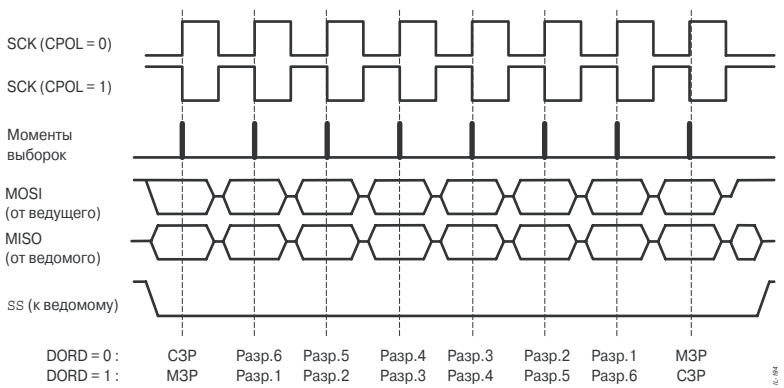


Рис. 2.108. Передача данных при CPHA = «0»

Частота тактового сигнала SCK и, соответственно, скорость передачи данных по интерфейсу определяется состоянием разрядов SPR1:SPR0

регистра SPCR и разряда SPI2X регистра SPSR (**Табл. 2.137**). Разумеется, речь идет о микроконтроллере, работающем в режиме «Master», т. к. именно он является источником тактового сигнала. Для устройства, находящегося в режиме «Slave», состояние этих разрядов безразлично.

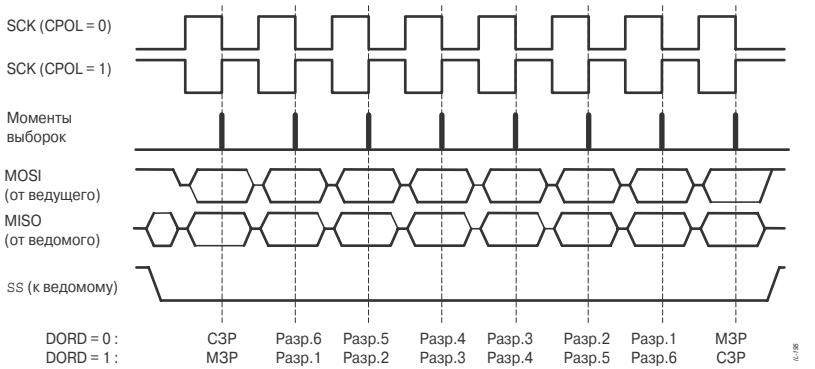


Рис. 2.109. Передача данных при CPHA = «1»

Таблица 2.137. **Задание частоты тактового сигнала SCK**

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

Примечание: f_{CLK} — тактовая частота микроконтроллера.

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

17.4. Использование вывода $\overline{SS}$

Вообще говоря, этот вывод предназначен для выбора активного ведомого устройства и в режиме «Slave» всегда является входом. При подаче на него напряжения НИЗКОГО уровня модуль SPI активируется и вывод MOSI переключается в режим вывода данных (если это задано пользователем).

Остальные выводы модуля SPI являются в этом режиме входами. А при подаче на вывод $\overline{SS}$ напряжения ВЫСОКОГО уровня **все** выводы модуля SPI переключаются в режим ввода данных. При этом модуль переходит в неактивное состояние и прием данных не производится. Как правило, в этом состоянии программа изменяет содержимое регистра данных.

Следует помнить, что каждый раз, когда на вывод $\overline{SS}$ подается напряжение ВЫСОКОГО уровня, происходит сброс модуля SPI. Соответственно, если изменение состояния этого вывода произойдет во время передачи данных, и прием, и передача немедленно прекратятся, а передаваемый и принимаемый байты будут потеряны.

Если же микроконтроллер находится в режиме «Master» (разряд MSTR регистра SPCR установлен в «1»), направление передачи данных через вывод $\overline{SS}$ определяется пользователем. Если вывод сконфигурирован как выход, он работает как линия вывода общего назначения и не влияет на работу модуля SPI. Как правило, в этом случае он используется для управления выводом $\overline{SS}$ микроконтроллера, работающего в режиме «Slave».

Если же вывод сконфигурирован как вход, то для обеспечения нормальной работы модуля SPI, на него должно быть подано напряжение ВЫСОКОГО уровня. Подача на этот вход напряжения НИЗКОГО уровня от какой-либо внешней схемы будет воспринята модулем SPI как выбор данного микроконтроллера в качестве ведомого, и соответственно, начало передачи ему данных. Во избежание конфликта на шине модуль SPI в таких случаях выполняет следующие действия:

1. Флаг MSTR регистра SPCR сбрасывается, и микроконтроллер переключается в режим «Slave». Как следствие, выводы MOSI и SCK начинают функционировать как входы.
2. Устанавливается флаг SPIF регистра SPSR, генерируя запрос на прерывание от SPI. Если прерывания от SPI разрешены и флаг I регистра SREG установлен в «1», происходит запуск подпрограммы обработки прерывания.

Таким образом, если ведущий микроконтроллер использует передачу данных, управляемую прерыванием, и существует вероятность подачи на вход $\overline{SS}$ напряжения НИЗКОГО уровня, в подпрограмме обработки прерывания от SPI обязательно должна осуществляться проверка состояния флага MSTR. При обнаружении сброса этого флага он должен быть программно установлен обратно в «1» для обратного перевода микроконтроллера в режим «Master».

Глава 18. **Последовательный двухпроводный интерфейс**

18.1. Общие сведения

Модуль двухпроводного последовательного интерфейса (Two-wire Serial Interface, TWI) входит в состав следующих микроконтроллеров семейства Mega: ATmega8x, ATmega16x, ATmega163x, ATmega32x, ATmega323x и ATmega64x/128x. Данный интерфейс является полным аналогом базовой версии интерфейса I^2C фирмы «Philips». Интерфейс TWI позволяет объединить вместе до 128 различных устройств с помощью двунаправленной шины, состоящей всего из двух линий: линии тактового сигнала (SCL) и линии данных (SDA). Единственными дополнительными элементами для реализации шины являются два подтягивающих резистора, по одному на каждую линию (**Рис. 2.110**).

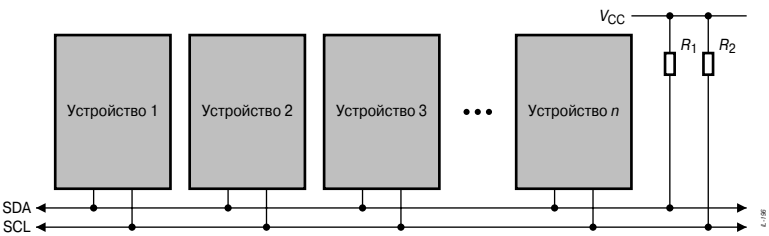


Рис. 2.110. Соединение устройств с помощью шины TWI

Шинные формирователи всех TWI-совместимых устройств выполняются по схеме с открытым коллектором (стоком), что позволяет реализовать функцию «монтажное И». Соответственно, НИЗКИЙ уровень на линии устанавливается тогда, когда одно или более устройств выставляют на линию сигнала лог. 0, а ВЫСОКИЙ уровень на линии устанавливается тогда, когда все устройства, подключенные к ней, устанавливают свои выходы в третье состояние.

При последующем рассмотрении модуля TWI будут часто встречаться некоторые термины. Их описание приведено в **Табл. 2.138**.

Таблица 2.138. **Термины, используемые при описании модуля TWI**

Термин	Описание
Ведущий (Master)	Устройство, инициирующее и завершающее передачу данных по шине, а также генерирующее тактовый сигнал SCL
Ведомый (Slave)	Устройство, адресуемое ведущим
Передатчик (Transmitter)	Устройство, выдающее данные на шину
Приемник (Reciever)	Устройство, считывающее данные с шины

18.2. Принципы обмена данными по шине TWI

Поскольку шина TWI является последовательной, все данные передаются по ней (по линии SDA) поразрядно. Каждый передаваемый разряд сопровождается импульсом на линии тактового сигнала SCL. Причем сигнал на линии SDA должен быть стабильным все то время, пока на шине SCL присутствует сигнал лог. 1 (**Рис. 2.111**). Единственным исключением из этого правила являются два особых состояния шины TWI — СТАРТ и СТОП.

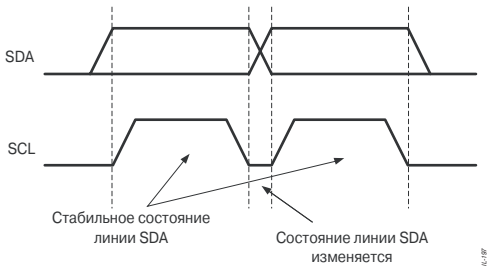


Рис. 2.111. Корректная выдача данных на шину SDA

Состояния СТАРТ и СТОП формируются ведущим в начале и в конце передачи данных соответственно. Между этими состояниями шина считается занятой, и другие ведущие не должны пытаться управлять ею.

Если ведущий хочет начать передачу нового блока данных без потери/восстановления контроля над шиной, он может сформировать состояние СТАРТ до формирования состояния СТОП. Формируемое таким образом состояние называется «повторный СТАРТ» (ПОВСТАРТ), а шина считается занятой до последующего формирования состояния СТОП. По-

скольку такое поведение ничем не отличается от поведения после формирования состояния СТАРТ, в дальнейшем оба эти состояния (СТАРТ и ПОВСТАРТ) будут обозначаться как СТАРТ, кроме тех случаев, где их необходимо различать.

Состояния СТАРТ и СТОП формируются путем изменения уровня сигнала на линии SDA при ВЫСОКОМ уровне на линии SCL. Состоянию СТАРТ соответствует смена уровня с «1» на «0», а состоянию СТОП — наоборот, с «0» на «1» (**Рис. 2.112**).

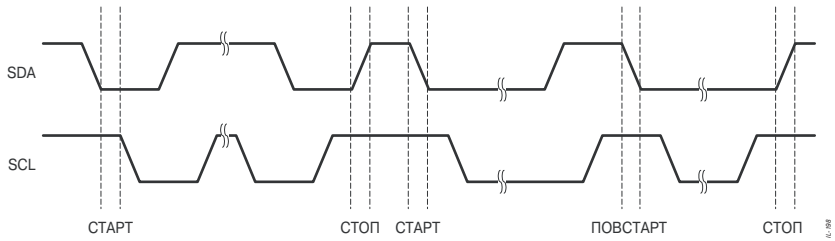


Рис. 2.112. Состояния СТАРТ, СТОП и ПОВСТАРТ

Заметим, что протокол интерфейса TWI позволяет подключать к шине несколько ведущих устройств (режим Multi-Master). При этом могут возникать различные проблемы, одной из которых является несовпадение частот тактовых сигналов, генерируемых разными ведущими.

Задача синхронизации решается очень просто благодаря подсоединению всех устройств к линии SCL по схеме «монтажное И». В результате длительность тактовых импульсов на линии SCL определяется тактовым сигналом с наименьшей длительностью импульсов. А пауза между тактовыми импульсами, наоборот, определяется тактовым сигналом с наибольшей паузой между импульсами. Помимо этого, все ведущие контролируют уровень, присутствующий на линии SCL, и определяют момент начала отсчета импульса или паузы тактового сигнала по соответствующему изменению сигнала (**Рис. 2.113**).

Другой задачей, которую приходится решать при подключении к шине нескольких ведущих устройств, является задача распределения приоритетов, в случае если два и более ведущих одновременно пытаются начать передачу. При возникновении такой ситуации передачу может осуществить только один ведущий, остальные же должны переключиться в режим ведомого. Причем передаваемые данные во время распределения приоритетов не должны искажаться.

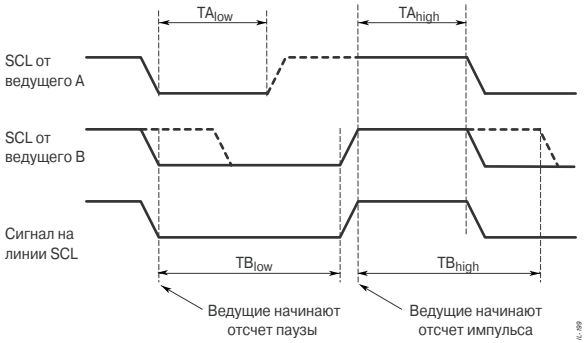


Рис. 2.113. Синхронизация сигнала SCL нескольких ведущих

Для решения описанной задачи все ведущие устройства после выдачи данных на линию SDA контролируют ее состояние. Если состояние линии отличается от того, в которое ее переводил ведущий, он теряет приоритет (**Рис. 2.114**). Еще раз напомним, что такое возможно только в случае выдачи им сигнала ВЫСОКОГО уровня во время выдачи другими ведущими (получившими приоритет) сигнала НИЗКОГО уровня. Ведущий, потерявший приоритет, должен немедленно переключиться в режим ведомого и проверить, не был ли он адресован каким-либо из ведущих, получивших приоритет. При этом он должен продолжать удерживать на линии SDA сигнал ВЫСОКОГО уровня. Однако он может продолжать генерировать тактовый сигнал до окончания передачи текущего пакета.

Процесс распределения приоритетов продолжается до тех пор, пока на шине не останется только один ведущий. В случае если несколько ведущих пытаются адресовать одного и того же ведомого, процесс распределения приоритетов продолжается и во время передачи пакета данных. Отсюда следует, в частности, что все циклы обмена должны содержать одинаковое количество пакетов данных, в противном случае результат процесса распределения приоритетов будет неопределенным.

При этом следует помнить, что процесс распределения приоритетов не должен выполняться между:

- состоянием ПОВСТАРТ и передачей разряда данных;
- состоянием СТОП и передачей разряда данных;
- состоянием ПОВСТАРТ и СТОП.

Ответственность за невозникновение перечисленных ситуаций ложится на программное обеспечение устройств, подключенных к шине.

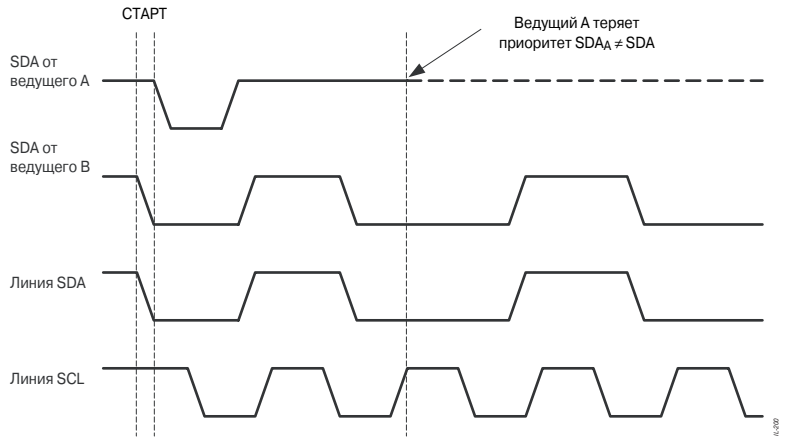


Рис. 2.114. Распределение приоритетов между двумя ведущими

При передаче данных по шине TWI вместе с ними передается определенная служебная информация. Совокупность данных и соответствующей служебной информации называется пакетом. Различают адресные пакеты и пакеты данных.

Формат адресного пакета

Все адресные пакеты, передаваемые по шине TWI, имеют длину 9 бит. Пакет включает 7-разрядный адрес (первым передается старший разряд), управляющий бит R/W и бит квитирования ACK (**Рис. 2.115**). Значение управляющего бита R/W определяет направление передачи данных по шине. Сброшенный в «0» бит означает передачу данных, а установленный в «1» — запрос данных (чтение). Пакеты со сброшенным и установленным управляющим битом обозначаются соответственно SLA-W и SLA-R.

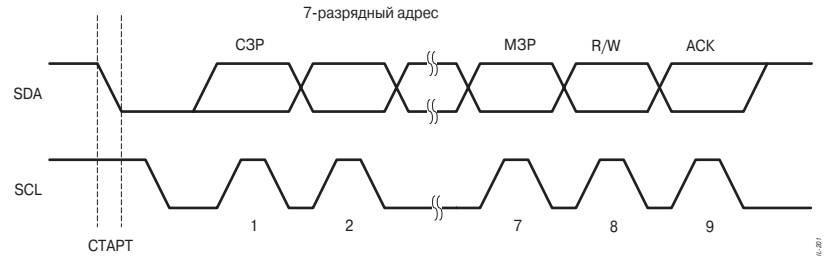


Рис. 2.115. Формат адресного пакета

При распознавании ведомым своего адреса он должен сформировать подтверждение путем выдачи на линию SDA сигнала НИЗКОГО уровня во время 9-го тактового импульса (ACK). Если ведомый по каким-либо причинам не может обслужить запрос ведущего, он должен во время 9-го тактового импульса удерживать на линии сигнал ВЫСОКОГО уровня (NACK).

Ведомым устройствам могут быть назначены любые адреса за исключением нулевого адреса и адресов из диапазона «1111000...1111111». Нулевой адрес зарезервирован для реализации так называемых общих вызовов, а адреса формата «1111xxx» должны быть зарезервированы для использования в дальнейшем.

Общий вызов используется, когда ведущий хочет передать всем ведомым, подключенным к шине, одно и то же сообщение. При приеме адресного пакета с нулевым адресом и сброшенным управляющим битом (запрос на передачу) все ведомые устройства, подключенные к шине, должны сформировать подтверждение. Исключение составляют лишь те устройства, в которых распознавание общего вызова по каким-либо причинам запрещено. Соответственно, последующие пакеты данных, посылаемые ведущим, будут получены всеми ведомыми устройствами, распознавшими адрес общего вызова.

Очевидно, что общий вызов с установленным управляющим разрядом (запрос на чтение) не имеет смысла, поскольку различные устройства не могут одновременно осуществлять передачу данных по шине.

Формат пакета данных

Все пакеты данных, передаваемые по шине TWI, тоже имеют длину 9 бит. Пакет состоит из байта данных (первым передается старший разряд) и бита квитирования ACK (**Рис. 2.116**). Генерация тактового сигнала и формирование состояний СТАРТ и СТОП осуществляется ведущим. Приемник, в свою очередь, должен после приема каждого байта формировать подтверждение (ACK) путем выдачи на линию SDA сигнала НИЗКОГО уровня во время 9-го тактового импульса. Если приемник получил последний байт или по каким-либо другим причинам не может продолжать прием данных, он должен во время 9-го тактового импульса удерживать на линии сигнал ВЫСОКОГО уровня (NACK). Не получив подтверждения от приемника, ведущий может прекратить передачу данных, сформировав состояние СТОП.

На практике каждый цикл обмена по шине TWI состоит из следующих этапов (**Рис. 2.117**):

1) формирование состояния СТАРТ;
2) передача адресного пакета SLA+R/W;
3) передача одного или нескольких пакетов данных;
4) формирование состояния СТОП.

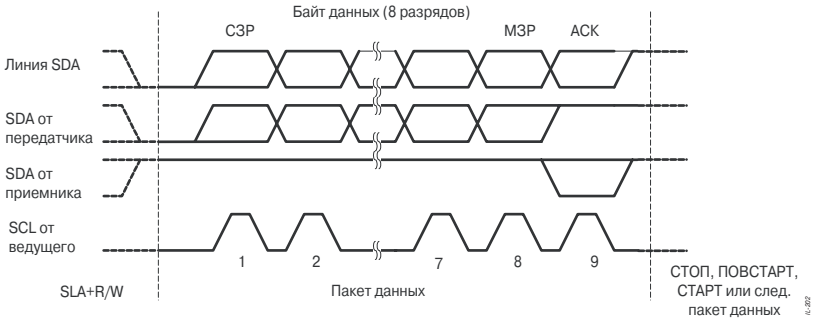


Рис. 2.116. Формат пакета данных

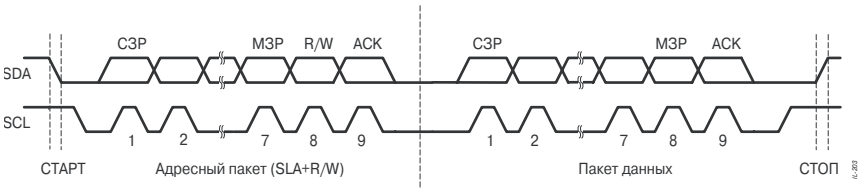


Рис. 2.117. Типичный цикл обмена

Скорость обмена по шине TWI задается ведущим, т. к. именно он генерирует тактовые импульсы. Однако, если ведомый не может принимать данные с такой скоростью или ему просто требуется время на обработку данных между приемом пакетов, он может увеличить паузу между тактовыми импульсами, удерживая на линии SCL сигнал НИЗКОГО уровня. На длительность тактовых импульсов это не влияет.

18.3. Обзор модуля TWI

Структурная схема модуля TWI приведена на **Рис. 2.118**. Как видно из рисунка, модуль включает в себя несколько блоков, каждый из которых выполняет определенные функции.

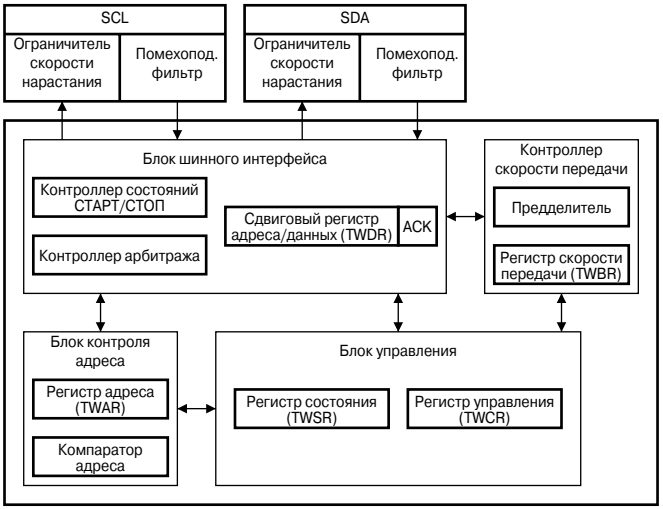


Рис. 2.118. Структурная схема модуля TWI

Взаимодействие программы с модулем TWI осуществляется посредством пяти регистров ввода/вывода. Названия и назначение этих регистров, а также их положение в адресном пространстве ввода/вывода всех моделей приведено в **Табл. 2.139**.

Таблица 2.139. **Регистры ввода/вывода модуля TWI**

Регистр	ATmega8x	ATmega16x	ATmega163x	ATmega32x	ATmega323x	ATmega64x ATmega128x	Назначение
TWBR	$00 ($20)	$00 ($20)	$00 ($20)	$00 ($20)	$00 ($20)	($70)	Регистр скорости передачи
TWSR	$01 ($21)	$01 ($21)	$01 ($21)	$01 ($21)	$01 ($21)	($71)	Регистр состояния
TWAR	$02 ($22)	$02 ($22)	$02 ($22)	$02 ($22)	$02 ($22)	($72)	Регистр адреса
TWDR	$03 ($23)	$03 ($23)	$03 ($23)	$03 ($23)	$03 ($23)	($73)	Регистр данных
TWCR	$36 ($56)	$36 ($56)	$36 ($56)	$36 ($56)	$36 ($56)	($74)	Регистр управления

Рассмотрим подробнее составные части модуля TWI.

Выводы SCL и SDA

Используются для подключения модуля к одноименным линиям шины TWI. В соответствии со спецификацией интерфейса TWI выходные

каскады, подключенные к выводам, содержат ограничители скорости нарастания сигнала, а входные каскады содержат фильтр, подавляющий паразитные импульсы длительностью менее 50 нс.

Оба вывода являются линиями портов ввода/вывода микроконтроллеров (**Табл. 2.140**).

Таблица 2.140. **Выводы микроконтроллеров, используемые модулем TWI**

Название	ATmega8x	ATmega16x	ATmega163x	ATmega32x	ATmega323x	ATmega64x ATmega128x	Назначение
SDA	PC4	PC1	PC1	PC1	PC1	PD1	Линия данных
SCL	PC5	PC0	PC0	PC0	PC0	PD0	Линия тактового сигнала

При использовании указанных выводов микроконтроллера модулем TWI к ним, как и при обычном их использовании, можно подключить внутренние подтягивающие резисторы. Это позволит в ряде случаев обойтись без внешних подтягивающих резисторов, требуемых спецификацией интерфейса TWI.

Контроллер скорости передачи (Bit Rate Generator)

Этот блок задает период следования импульсов на линии SCL при работе микроконтроллера в режиме ведущего. В моделях ATmega163x и ATmega323x управление скоростью передачи (частотой сигнала SCL) осуществляется с помощью регистра TWBR. Частота формируемого сигнала SCL (при работе устройства в режиме ведущего) для этих моделей определяется выражением $f_{SCL} = f_{CLK}/(16 + 2TWBR)$.

В моделях ATmega8x, ATmega16x, ATmega32x, ATmega64x, ATmega128x вместе с регистром TWBR используются два младших разряда (TWPS1:TWPS0) регистра TWSR. Для этих моделей частота формируемого сигнала SCL определяется выражением $f_{SCL} = f_{CLK}/(16 + 2TWBR \cdot 4^{TWPS})$.

В приведенных формулах f_{CLK} — тактовая частота процессора, *TWBR* — значение, записанное в регистре TWBR, TWPS — значение разрядов TWPS1:TWPS0 регистра TWSR.

И регистр TWBR, и разряды TWPS1:TWPS0 регистра TWSR доступны как для чтения, так и для записи в любой момент времени. При работе в режиме ведомого содержимое указанных разрядов безразлично, однако тактовая частота микроконтроллера в этом случае должна быть, как минимум, в 16 раз выше частоты сигнала SCL.

Еще раз напоминаем, что любое устройство, подключенное к шине TWI, может увеличивать длительность паузы между тактовыми импульсами, снижая, таким образом, скорость передачи данных по шине.

Блок шинного интерфейса (Bus Interface Unit)

В состав этого блока входят два узла:

- контроллер состояний СТАРТ/СТОП формирует и обнаруживает состояния СТАРТ, ПОВСТАРТ и СТОП. Мониторинг состояния шины ведется даже при нахождении микроконтроллера в «спящем» режиме. Благодаря этому обеспечивается при необходимости выход микроконтроллера из «спящего» режима при адресации его каким-либо ведущим;
- контроллер арбитража определяет наличие конфликтов на шине при работе микроконтроллера в режиме ведущего. В случае потери устройством приоритета контроллер информирует об этом блок управления, который производит необходимые действия и формирует соответствующие коды состояния.

Кроме того, в состав блока входит сдвиговый регистр адреса/данных TWDR, который содержит данные передаваемого или принимаемого пакета. Одновременно с выдвиганием содержимого регистра на шину данные с нее вдвигаются в этот регистр. Таким образом, почти всегда, за исключением момента выхода микроконтроллера из «спящего» режима, в регистре TWDR содержится последний байт, имевшийся на шине. При включении питания все разряды этого регистра устанавливаются в «1». При этом инициализация регистра пользователем может быть осуществлена только после генерации первого прерывания.

Помимо регистра TWDR в составе блока имеется специальный регистр, один из разрядов которого содержит значение переданного или принятого бита подтверждения. При приеме состояние этого бита задается одним из разрядов регистра управления TWCR, а при передаче состояние полученного бита подтверждения может быть определено по коду, находящемуся в регистре состояния TWSR.

Блок контроля адреса (Address Match Unit)

Блок контроля адреса проверяет принятый адрес на соответствие значению, находящемуся в старших семи разрядах регистра адреса TWAR. Также он проверяет наличие общих вызовов, если разрешено их распознавание. При обнаружении корректного адреса он информирует об этом блок управления, который формирует или не формирует подтверждение получения адресного пакета. Контроль адресных пакетов осуществляется блоком даже при нахождении микроконтроллера в «спящем» режиме, что позволяет

перевести микроконтроллер в рабочий режим в случае адресации его ведущим устройством.

Формат регистра TWAR показан на **Рис. 2.119**, а описание его разрядов приведено в **Табл. 2.141**.

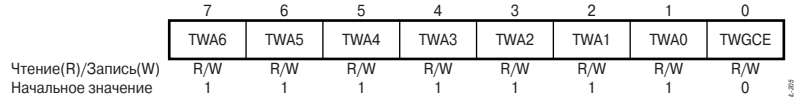


Рис. 2.119. Формат регистра TWAR

Таблица 2.141. **Разряды регистра адреса TWAR**

Разряд	Название	Описание
7...1	TWA6...TWA0	**Адрес устройства.** В этих разрядах содержится адрес, на который устройство будет отзываться при работе в режиме ведомого. При работе устройства в режиме ведущего содержимое этих разрядов безразлично
0	TWGCE	**Разрешение распознавания общих вызовов.** Если этот разряд установлен в «1», устройство будет отзываться на общие вызовы (пакеты с адресом $00) так же, как и на вызовы с адресом, находящимся в разрядах TWA6...0 регистра. Если разряд сброшен в «0», распознавание общих вызовов запрещено

Блок управления (Control Unit)

Этот блок осуществляет управление всем модулем TWI в соответствии с установками регистра управления TWCR и информацией, поступающей к нему от остальных блоков модуля. При наступлении определенных событий, указанных ниже, блок управления формирует в регистре состояния TWSR код статуса, соответствующий событию и устанавливает флаг запроса на прерывание TWINT регистра TWCR. До момента сброса этого флага на линии SCL удерживается НИЗКИЙ уровень, приостанавливая тем самым передачу данных по шине.

Формирование запроса на прерывание осуществляется при возникновении следующих событий:

- окончание формирования состояния СТАРТ/ПОВСТАРТ;
- окончание передачи адресного пакета (SLA+R/W);
- окончание передачи байта адреса;
- потеря устройством приоритета;
- адресация устройства или наличие общего вызова;
- окончание приема байта данных;
- возникновение ошибок на шине, обусловленных недопустимыми условиями формирования состояний СТАРТ/СТОП.

Формат регистра TWCR показан на **Рис. 2.120**, а описание его разрядов приведено в **Табл. 2.142**. Формат регистра TWSR показан на **Рис. 2.121**, а описание его разрядов приведено в **Табл. 2.143**.

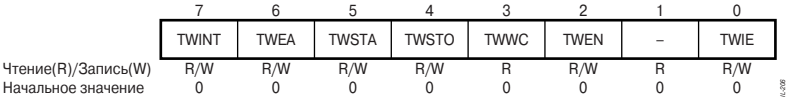


Рис. 2.120. Формат регистра TWCR

Таблица 2.142. **Разряды регистра управления TWCR**

Разряд	Название	Описание
7	TWINT	**Флаг прерывания от модуля TWI**. Этот флаг устанавливается аппаратно после выполнения очередной операции, когда модуль ожидает отклика со стороны программы. Если установлены флаги I регистра SREG и TWIE регистра TWCR, генерируется прерывание и осуществляется вызов соответствующего обработчика. Пока флаг TWINT установлен, на линии SCL удерживается сигнал НИЗКОГО уровня. Сброс флага может быть осуществлен только записью в него лог. 1
6	TWEA	**Разрешение бита подтверждения.** Разряд TWEA управляет формированием бита подтверждения. Если этот разряд установлен в «1», то устройство формирует сигнал подтверждения, когда это необходимо. При сбросе разряда в «0» устройство виртуально отключается от шины TWI, т. е. бит подтверждения не формируется
5	TWSTA	**Флаг состояния СТАРТ.** При записи в разряд TWSTA лог. 1 модуль проверяет состояние шины TWI и, если шина свободна, формирует состояние СТАРТ. Если шина занята, модуль TWI ожидает появления состояния СТОП, и только после этого формирует состояние СТАРТ. Флаг сбрасывается аппаратно по окончании формирования состояния СТАРТ
4	TWSTO	**Флаг состояния СТОП.** В режиме ведущего установка флага TWSTO в «1» приводит к формированию на шине состояния СТОП. Флаг сбрасывается аппаратно по окончании формирования состояния СТОП. Установка флага TWSTO в режиме ведомого может использоваться для выхода из ошибочных ситуаций. После записи в этот разряд лог. 1 модуль TWI возвращается в режим неадресованного ведомого, а выводы SCL и SDA устанавливаются в третье состояние. Состояние СТОП не формируется
3	TWWC	**Флаг конфликта записи.** Флаг устанавливается в «1» при попытке записи в регистр TWDR, когда флаг прерывания TWINT сброшен. Флаг сбрасывается при записи в регистр TWDR, когда флаг прерывания TWINT установлен
2	TWEN	**Разрешение работы модуля TWI.** Разряд TWEN управляет функционированием модуля TWI. При записи в этот разряд лог. 1 модуль TWI включается и берет на себя управление контактами ввода/вывода микроконтроллера, соответствующих выводам SCL и SDA. При сбросе разряда TWEN в 0 модуль TWI выключается
1	—	Зарезервирован, читается как «0»
0	TWIE	**Разрешение прерывания от модуля TWI.** Если в этом разряде записана лог. 1 и флаг I регистра SREG также установлен в «1», то прерывание от модуля TWI разрешено

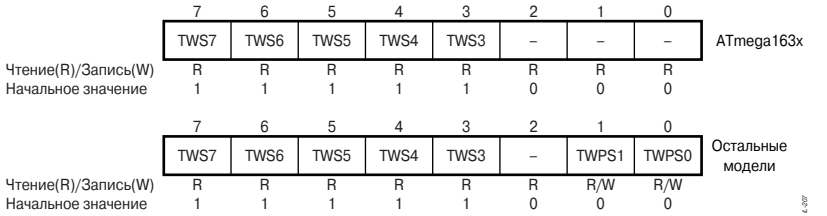


Рис. 2.121. Формат регистра TWSR

Таблица 2.143. **Разряды регистра состояния TWSR**

Разряд	Название	Описание
7...3	TWS7...TWS3	**Состояние модуля TWI.** Значение, содержащееся в этих разрядах, отражает состояние узлов модуля и шины TWI. Возможные коды статуса будут описаны при дальнейшем рассмотрении модуля TWI. Разряды TWS7...3 доступны только для чтения
2	—	Зарезервирован, читается как «0»
1, 0	TWPS1:TWPS0	**Коэффициент деления предделителя контроллера скорости передачи.** Состояние этих разрядов определяет коэффициент деления предделителя контроллера скорости передачи, управляя частотой генерируемого сигнала SCL (см. описание контроллера скорости передачи в начале подраздела)

В регистре TWSR моделей ATmega163x и ATmega323x эти разряды не используются (зарезервированы) и читаются как «0».

18.4. Взаимодействие прикладной программы с модулем TWI

Взаимодействие прикладных программ с модулем TWI базируется на использовании прерывания от модуля, которое возникает после каждого события, произошедшего на шине (прием байта, формирование состояний СТАРТ/СТОП и т. п.). Соответственно, во время передачи данных по шине программа может выполнять другие задачи. Если прерывание от модуля TWI по каким-либо причинам использовать нельзя, его можно запретить. В этом случае программа должна будет постоянно следить за состоянием флага TWINT для реагирования на события, происходящие на шине.

Установка флага TWINT регистра TWCR означает, что модуль TWI закончил выполнение очередной операции и ожидает реакции программы.

В старших пяти разрядах (TWS7:0) регистра TWSR при этом формируется некоторое значение, характеризующее текущее состояние шины TWI. Соответственно, программа должна проанализировать это значение и задать дальнейшее поведение модуля TWI, манипулируя содержимым регистров TWCR и TWDR.

На **Рис. 2.122** показано взаимодействие прикладной программы с модулем TWI на простейшем примере передачи одного байта данных от ведущего к ведомому.

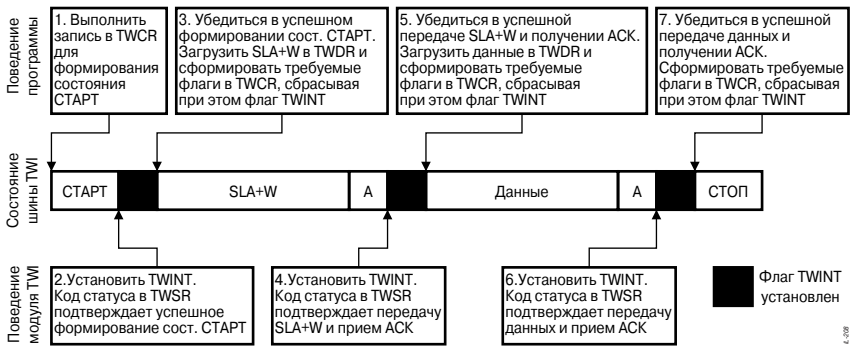


Рис. 2.122. Пример взаимодействия программы с модулем TWI

Передача данных происходит в следующей последовательности:

1. Первой операцией при передаче данных по шине TWI является формирование состояния СТАРТ. Для этого следует записать определенное значение в регистр TWCR, в соответствии с которым модуль TWI сформирует на шине состояние СТАРТ. При этом разряд записываемого значения, соответствующий флагу TWINT, должен быть установлен в «1» для сброса этого флага. Формирование состояния СТАРТ начнется сразу же после сброса флага TWINT.
2. После формирования состояния СТАРТ устанавливается флаг TWINT. Число, находящееся в регистре состояния TWSR, индицирует результат выполнения этой операции.
3. Необходимо удостовериться в успешном формировании состояния СТАРТ, проверив содержимое регистра TWSR. Если код статуса соответствует ожидаемому, следует загрузить в регистр TWDR содержимое пакета SLA+W и сформировать в регистре TWCR команду на передачу пакета (не забывая сбросить при этом флаг TWINT). Передача адресного пакета начнется сразу же после сброса флага.

4. По окончании передачи адресного пакета устанавливается флаг TWINT. Код статуса, находящийся в регистре TWCR, индицирует об успешной передаче пакета, а также о получении подтверждения от ведомого устройства.
5. Необходимо удостовериться в успешной передаче пакета и получении подтверждения, проверив содержимое регистра TWSR. Если код статуса соответствует ожидаемому, следует загрузить данные в регистр TWDR, а затем сформировать в регистре TWCR команду на передачу пакета (не забывая сбросить при этом флаг TWINT). Передача пакета данных начнется сразу же после сброса флага.
6. По окончании передачи пакета данных устанавливается флаг TWINT. Код статуса, находящийся в регистре TWCR, индицирует об успешной передаче пакета, а также о получении подтверждения от ведомого устройства.
7. Необходимо удостовериться в успешной передаче пакета и получении подтверждения, проверив содержимое регистра TWSR. Если код статуса соответствует ожидаемому, следует записать в регистр TWCR значение (не забывая сбросить при этом флаг TWINT), в соответствии с которым модуль TWI сформирует на шине состояние СТОП. Формирование состояния СТОП начнется сразу же после сброса флага TWINT.

Из сказанного видно, что любой этап взаимодействия прикладной программы с модулем TWI состоит из трех частей:

- после завершения модулем выполнения какой-либо операции, он устанавливает флаг TWINT регистра TWCR и ожидает реакции программы. Пока флаг установлен в «1», на линии SCL удерживается НИЗКИЙ уровень;
- после установки флага TWINT пользователь должен занести в регистры модуля значения, соответствующие следующему этапу обмена;
- после обновления содержимого регистров модуля, пользователь должен сформировать в регистре TWCR команду на выполнение следующего этапа обмена. При загрузке в регистр нового значения необходимо сбросить флаг TWINT записью в него лог. 1. После сброса флага модуль начнет выполнение операции, определяемой содержимым регистра TWCR.

Возможные значения кодов статуса, о которых упоминается в примере, будут приведены далее при описании конкретных режимов работы модуля TWI. Кроме того, имеется два кода состояния ($F8 и $00), не связанные с каким-либо режимом работы (**Табл. 2.144**).

Состояние с кодом $F8 является промежуточным. Наличие этого кода означает отсутствие какой-либо информации в связи с тем, что флаг TWINT не установлен.

Код статуса $00 сигнализирует об ошибке на шине. Такая ошибка возникает при формировании ведущим состояний СТАРТ или СТОП в неположенном месте, например, во время передачи байта адреса, байта данных или бита подтверждения. Для выхода из таких ситуаций необходимо записать лог. 1 в разряды TWSTO и TWINT. В результате модуль перейдет в режим неадресованного ведущего, освободит линии SDA и SCL и сбросит флаг TWSTO. Состояние СТОП в этой ситуации сформировано не будет.

Таблица 2.144. **Коды статуса общего назначения**

Код статуса	Состояние шины и модуля TWI	Действия программы					Следующее действие, выполняемое модулем TWI
		в/из TWDR	в регистр TWCR				
			STA	STO	TWINT	TWEA	
$F8	Нет информации; TWINT = «0»	Нет действий	Нет действий				Ждать установки флага TWINT
$00	Ошибка на шине в результате некорректного формирования состояния СТАРТ или СТОП	Нет действий	0	1	1	X	Все действия выполняются аппаратно. Шина освобождается, а флаг TWSTO сбрасывается в «0»

18.5. Режимы работы модуля TWI

Модуль TWI, реализованный в микроконтроллерах семейства Mega, может работать в следующих режимах:

- ведущий передатчик (Master Transmitter);
- ведущий приемник (Master Receiver);
- ведомый передатчик (Slave Transmitter);
- ведомый приемник (Slave Receiver).

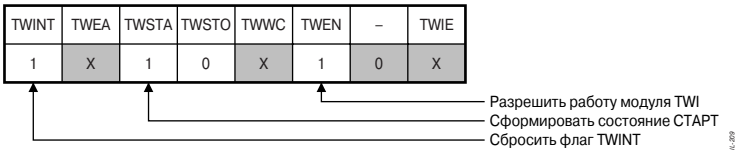
Выбор конкретного режима определяется логикой работы программы и, соответственно, выполняемыми действиями.

18.5.1. Режим «Ведущий передатчик»

В режиме «Ведущий передатчик» (Master Transmitter) осуществляется передача данных от ведущего устройства к ведомому. Для переключения устройства в режим ведущего модуль TWI должен сформировать на шине состояние СТАРТ. Формат адресного пакета, передаваемого затем, определяет в каком из режимов будет работать ведущий. При передаче пакета

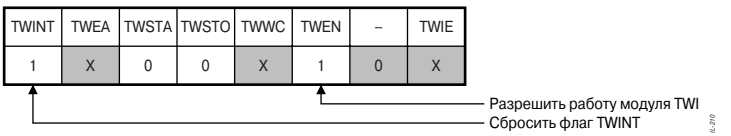
SLA+W модуль переходит в режим «Ведущий передатчик», а при передаче пакета SLA+R — в режим «Ведущий приемник».

Формирование состояния СТАРТ начнется после записи в регистр TWCR следующего значения:



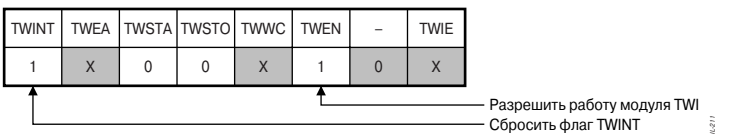
В результате записи указанного значения модуль TWI начнет контролировать состояние шины и сформирует состояние СТАРТ сразу же, как только она станет свободной.

По окончании формирования состояния СТАРТ устанавливается флаг TWINT; код статуса должен при этом иметь значение, равное $08 (см. **Табл. 2.144**). Для переключения модуля в режим «Ведущий передатчик» необходимо передать по шине пакет SLA+W. Для этого содержимое пакета загружается в регистр TWDR, а в регистр TWCR заносится следующее значение:

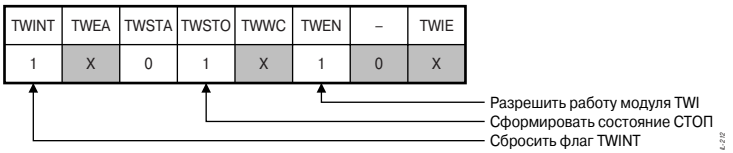


После передачи адресного пакета и приема бита подтверждения флаг TWINT снова устанавливается в «1». Код статуса на этом этапе может иметь одно из следующих значений: $18, $20 или $38. Какие действия необходимо предпринять при обнаружении того или иного кода, подробно рассмотрено в **Табл. 2.145**.

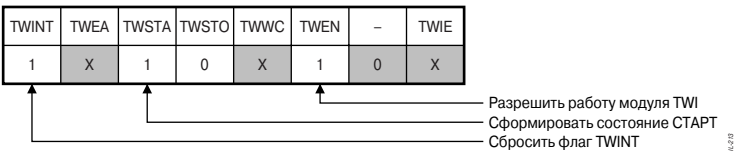
После передачи адресного пакета должны быть переданы пакеты данных. Значение байта данных загружается в регистр TWDR. Передача пакета данных начинается после записи в регистр TWCR следующего значения:



Описанная процедура используется для передачи всех пакетов данных. После передачи последнего байта данных, ведущий должен сформировать на шине состояние СТОП или ПОВСТАРТ. Формирование состояния СТОП начнется после записи в регистр TWCR следующего значения:



А для формирования состояния ПОВСТАРТ в регистр TWCR необходимо занести следующее значение:



После формирования на шине состояния ПОВСТАРТ (код статуса $10) ведущий может адресовать того же или другого ведомого не формируя состояния СТОП. Другими словами, использование состояния ПОВСТАРТ позволяет осуществлять смену ведомых устройств, а также переключаться между режимами «Ведущий передатчик» и «Ведущий приемник» без утери контроля над шиной.

Таблица 2.145. **Коды статуса для режима «Ведущий передатчик»**

Код статуса	Состояние шины и модуля TWI	Действия программы					Следующее действие, выполняемое модулем TWI
		в/из TWDR	в регистр TWCR				
			STA	STO	TWINT	TWEA	
$08	Было сформировано состояние СТАРТ	Загрузить SLA + W	X	0	1	X	Будет передан SLA + W. Будет получен ACK или NACK
$10	Было сформировано состояние ПОВСТАРТ	Загрузить SLA + W	X	0	1	X	Будет передан SLA + W. Будет получен ACK или NACK
		Загрузить SLA + R	X	0	1	X	Будет передан SLA + R. Модуль переключится в режим «Ведущий приемник»
$18	Был передан пакет SLA+W и принято подтверждение (ACK)	Загрузить данные	0	0	1	X	Будет передан байт данных. Будет получен ACK или NACK
		Нет действий	1	0	1	X	Будет сформировано состояние ПОВСТАРТ
		Нет действий	0	1	1	X	Будет сформировано состояние СТОП (флаг TWSTO будет сброшен)
		Нет действий	1	1	1	X	Будет сформировано состояние СТОП, а затем состояние СТАРТ (флаг TWSTO будет сброшен)

Продолжение таблицы 2.145

Код статуса	Состояние шины и модуля TWI	Действия программы					Следующее действие, выполняемое модулем TWI
		в/из TWDR	в регистр TWCR				
			STA	STO	TWINT	TWEA	
$20	Был передан пакет SLA+W, а подтверждение не было принято (NACK)	Загрузить данные	0	0	1	X	Будет передан байт данных. Будет получен ACK или NACK
		Нет действий	1	0	1	X	Будет сформировано состояние ПОВСТАРТ
		Нет действий	0	1	1	X	Будет сформировано состояние СТОП (флаг TWSTO будет сброшен)
		Нет действий	1	1	1	X	Будет сформировано состояние СТОП, а затем состояние СТАРТ (флаг TWSTO будет сброшен)
$28	Был передан пакет данных и принято подтверждение (ACK)	Загрузить данные	0	0	1	X	Будет передан байт данных. Будет получен ACK или NACK
		Нет действий	1	0	1	X	Будет сформировано состояние ПОВСТАРТ
		Нет действий	0	1	1	X	Будет сформировано состояние СТОП (флаг TWSTO будет сброшен)
		Нет действий	1	1	1	X	Будет сформировано состояние СТОП, а затем состояние СТАРТ (флаг TWSTO будет сброшен)
$30	Был передан пакет данных, а подтверждение не было принято (NACK)	Загрузить данные	0	0	1	X	Будет передан байт данных. Будет получен ACK или NACK
		Нет действий	1	0	1	X	Будет сформировано состояние ПОВСТАРТ
		Нет действий	0	1	1	X	Будет сформировано состояние СТОП (флаг TWSTO будет сброшен)
		Нет действий	1	1	1	X	Будет сформировано состояние СТОП, а затем состояние СТАРТ (флаг TWSTO будет сброшен)
$38	Потеря приоритета при передаче пакета адреса или данных	Нет действий	0	0	1	X	Устройство освободит шину и перейдет в режим неадресованного ведомого
		Нет действий	1	0	1	X	После освобождения шины будет сформировано состояние СТАРТ

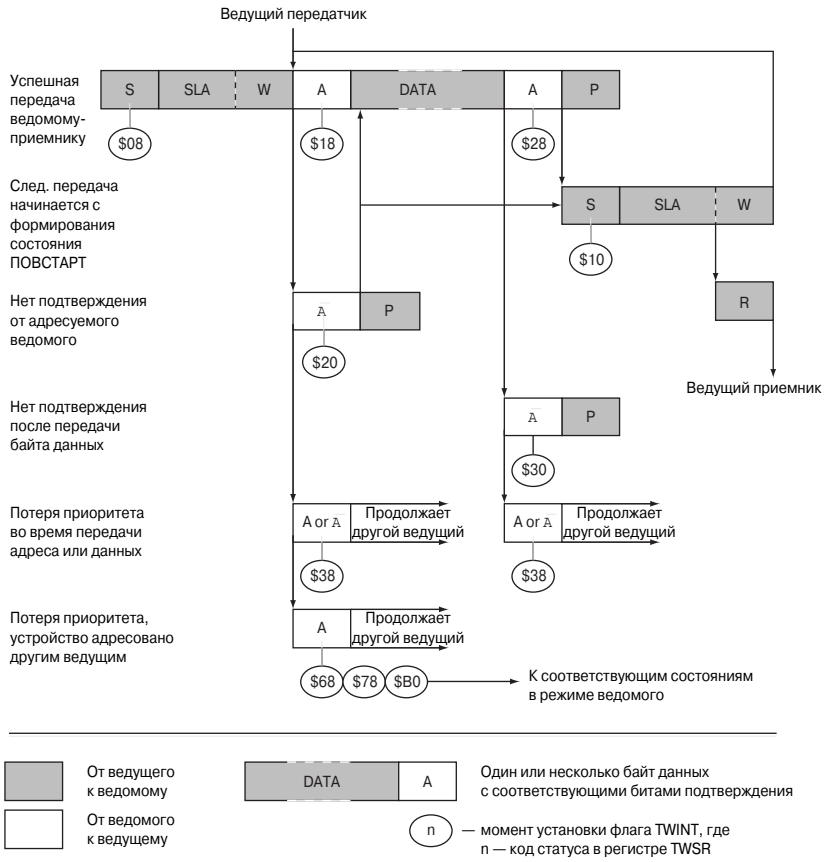
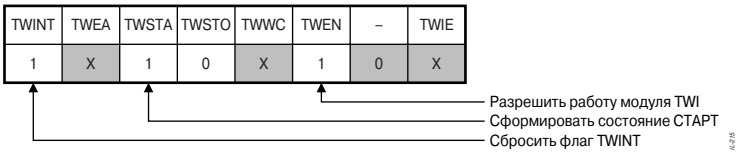


Рис. 2.123. Состояния модуля TWI в режиме «Ведущий передатчик»

18.5.2. Режим «Ведущий приемник»

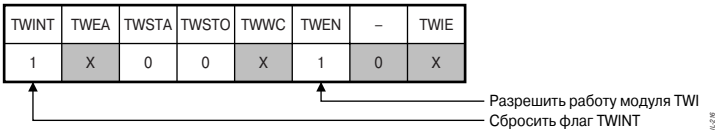
В режиме «Ведущий приемник» ведущий осуществляет прием данных от ведомого устройства. Как обычно, для переключения устройства в режим ведущего модуль TWI должен сформировать на шине состояние СТАРТ. Формат адресного пакета, передаваемого следом, определяет, в каком из режимов будет работать ведущий. При передаче пакета SLA+W модуль переходит в режим «Ведущий передатчик», а при передаче пакета SLA+R переходит в режим «Ведущий приемник».

Формирование состояния СТАРТ начинается после записи в регистр TWCR следующего значения:



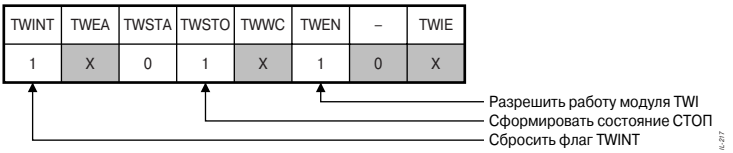
В результате записи указанного значения модуль TWI начнет контролировать состояние шины и сформирует состояние СТАРТ сразу же, как только она станет свободной.

По окончании формирования состояния СТАРТ устанавливается флаг TWINT; код статуса должен при этом иметь значение, равное $08 (**Табл. 2.146**). Для переключения модуля в режим «Ведущий передатчик» необходимо передать по шине пакет SLA+R. Для этого содержимое пакета загружается в регистр TWDR, а в регистр TWCR заносится следующее значение:



После передачи адресного пакета и приема бита подтверждения флаг TWINT снова устанавливается в «1». Полученный от ведомого байт данных находится при этом в регистре TWDR. Код статуса на этом этапе может иметь одно из следующих значений: $38, $40 или $48. Какие действия необходимо предпринять при обнаружении того или иного кода подробно рассмотрено в **Табл. 2.146**.

Описанная процедура используется для передачи всех пакетов данных. После приема последнего байта данных ведущий должен проинформировать об этом ведомого передатчика, послав сигнал неподтверждения (NACK). Затем ведущий должен сформировать на шине состояние СТОП или ПОВСТАРТ. Формирование состояния СТОП начнется после записи в регистр TWCR следующего значения:



Для формирования состояния ПОВСТАРТ в регистр TWCR необходимо занести следующее значение:

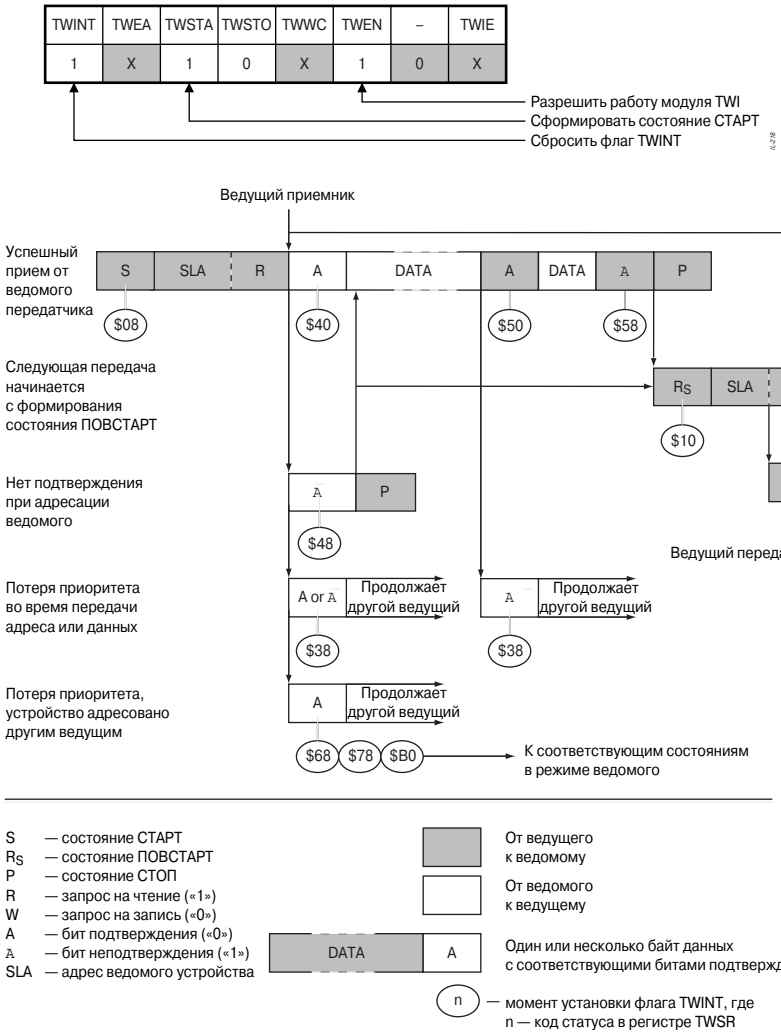


Рис. 2.124. Состояния модуля TWI в режиме «Ведущий приемник»

Как уже было сказано, после формирования на шине состояния ПОВСТАРТ (код статуса $10) ведущий может адресовать того же или друго-

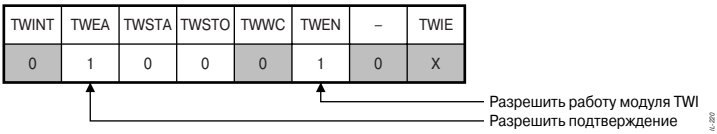
го ведомого, не формируя состояния СТОП. Другими словами, использование состояния ПОВСТАРТ позволяет осуществлять смену ведомых устройств, а также переключаться между режимами «Ведущий передатчик» и «Ведущий приемник» без утери контроля над шиной.

Таблица 2.146. **Коды статуса для режима «Ведущий приемник»**

Код статуса	Состояние шины и модуля TWI	Действия программы					Следующее действие, выполняемое модулем TWI
		в/из TWDR	в регистр TWCR				
			STA	STO	TWINT	TWEA	
$08	Было сформировано состояние СТАРТ	Загрузить SLA+R	X	0	1	X	Будет передан SLA+R. Будет получен ACK или NACK
$10	Было сформировано состояние ПОВСТАРТ	Загрузить SLA+R	X	0	1	X	Будет передан SLA+R. Будет получен ACK или NACK
		Загрузить SLA+W	X	0	1	X	Будет передан SLA+W. Модуль переключится в режим «Ведущий приемник»
$38	Потеря приоритета при передаче пакета адреса или данных	Нет действий	0	0	1	X	Устройство освободит шину и перейдет в режим неадресованного ведомого
		Нет действий	1	0	1	X	После освобождения шины будет сформировано состояние СТАРТ
$40	Был передан пакет SLA+R и принято подтверждение (ACK)	Нет действий	0	0	1	0	Будет принят байт данных и передано неподтверждение (NACK)
		Нет действий	0	0	1	1	Будет принят байт данных и передано подтверждение (ACK)
$48	Был передан пакет SLA+R и принято неподтверждение (NACK)	Нет действий	1	0	1	X	Будет сформировано состояние ПОВСТАРТ
		Нет действий	0	1	1	X	Будет сформировано состояние СТОП (флаг TWSTO будет сброшен)
		Нет действий	1	1	1	X	Будет сформировано состояние СТОП, а затем состояние СТАРТ (флаг TWSTO будет сброшен)
$50	Был принят байт данных и передано подтверждение (ACK)	Считать данные	0	0	1	0	Будет принят байт данных и передано неподтверждение (NACK)
		Считать данные	0	0	1	1	Будет принят байт данных и передано подтверждение (ACK)
$58	Был принят байт данных и передано неподтверждение (NACK)	Считать данные	1	0	1	X	Будет сформировано состояние ПОВСТАРТ
		Считать данные	0	1	1	X	Будет сформировано состояние СТОП (флаг TWSTO будет сброшен)
		Считать данные	1	1	1	X	Будет сформировано состояние СТОП, а затем состояние СТАРТ (флаг TWSTO будет сброшен)

18.5.3. Режим «Ведомый приемник»

В режиме «Ведомый приемник» ведомое устройство осуществляет прием данных от ведущего. Перед тем как переключить модуль в режим «Ведомый приемник», следует занести в старшие разряды регистра TWAR адрес устройства и в соответствии с логикой работы программы установить или сбросить младший разряд регистра (TWGCE). Затем необходимо записать в регистр TWCR следующее значение:



После инициализации регистров TWAR и TWCR модуль будет ожидать адресного пакета с адресом, указанным в регистре TWAR, или общего вызова (если его распознавание разрешено). Значение следующего за адресом бита направления определит режим, в который переключится модуль. Если этот бит будет сброшен в «0» (запись), модуль TWI переключится в режим «Ведомый приемник». В противном случае модуль переключится в режим «Ведомый передатчик». Напоминаем еще раз, что устройство также может самостоятельно переключиться в режим «Ведомый приемник» из режима ведущего при потере приоритета (см. описание кодов статуса $68 и $78 в **Табл. 2.147**).

После приема пакета SLA+W установится флаг TWINT и состояние обмена по шине можно будет, как обычно, определить по коду статуса. Возможные действия со стороны программы, соответствующие тому или иному коду статуса, указаны в **Табл. 2.147**.

Чтобы прервать поток данных, следует сбросить в «0» разряд TWEA регистра TWCR (это можно сделать и во время обмена, т. е. при сброшенном флаге TWINT). В результате после передачи следующего байта на линию SDA будет выдан сигнал неподтверждения, сигнализирующий ведущему о том, что ведомый не может больше осуществлять прием данных. Обработка адресных пакетов при сброшенном разряде TWEA прекращается, но может быть возобновлена в любой момент времени повторной установкой этого разряда.

Если адресация устройства произойдет при нахождении микроконтроллера в «спящем» режиме и разряд TWEA будет при этом установлен, модуль TWI переведет микроконтроллер в рабочий режим.

Таблица 2.147. **Коды статуса для режима «Ведомый приемник»**

Код статуса	Состояние шины и модуля TWI	Действия программы					Следующее действие, выполняемое модулем TWI
		в/из TWDR	в регистр TWCR				
			STA	STO	TWINT	TWEA	
$60	Был принят SLA+W с собственным адресом и послано подтверждение (ACK)	Нет действий	X	0	1	0	Будет принят байт данных и передано неподтверждение (NACK)
		Нет действий	X	0	1	1	Будет принят байт данных и передано подтверждение (ACK)
$68	Потеря приоритета в режиме ведущего во время передачи SLA+R/W; был принят SLA+W с собственным адресом и послано подтверждение (ACK)	Нет действий	X	0	1	0	Будет принят байт данных и передано неподтверждение (NACK)
		Нет действий	X	0	1	1	Будет принят байт данных и передано подтверждение (ACK)
$70	Был принят общий вызов и послано подтверждение (ACK)	Нет действий	X	0	1	0	Будет принят байт данных и передано неподтверждение (NACK)
		Нет действий	X	0	1	1	Будет принят байт данных и передано подтверждение (ACK)
$78	Потеря приоритета в режиме ведущего во время передачи SLA+R/W; был принят общий вызов и послано подтверждение (ACK)	Нет действий	X	0	1	0	Будет принят байт данных и передано неподтверждение (NACK)
		Нет действий	X	0	1	1	Будет принят байт данных и передано подтверждение (ACK)

Продолжение таблицы 2.147

Код статуса	Состояние шины и модуля TWI	Действия программы					Следующее действие, выполняемое модулем TWI
		в/из TWDR	в регистр TWCR				
			STA	STO	TWINT	TWEA	
$80	Устройство уже адресовано: был принят байт данных и послано подтверждение (ACK)	Считать данные	X	0	1	0	Будет принят байт данных и передано неподтверждение (NACK)
		Считать данные	X	0	1	1	Будет принят байт данных и передано подтверждение (ACK)
$88	Устройство уже адресовано: был принят байт данных и послано неподтверждение (NACK)	Считать данные	0	0	1	0	Переключиться в режим неадресованного ведомого; распознавание любых вызовов отключено
		Считать данные	0	0	1	1	Переключиться в режим неадресованного ведомого; разрешено распознавание SLA с собственным адресом; разрешено распознавание общих вызовов, если TWGCE = «1»
		Считать данные	1	0	1	0	Переключиться в режим неадресованного ведомого; распознавание любых вызовов запрещено; после освобождения шины будет сформировано состояние СТАРТ
		Считать данные	1	0	1	1	Переключиться в режим неадресованного ведомого; разрешено распознавание SLA с собственным адресом; разрешено распознавание общих вызовов, если TWGCE = «1»; после освобождения шины будет сформировано состояние СТАРТ
$90	Устройство уже адресовано (общий вызов): был принят байт данных и послано подтверждение (ACK)	Считать данные	X	0	1	0	Будет принят байт данных и передано неподтверждение (NACK)
		Считать данные	X	0	1	1	Будет принят байт данных и передано подтверждение (ACK)

Продолжение таблицы 2.147

Код статуса	Состояние шины и модуля TWI	Действия программы					Следующее действие, выполняемое модулем TWI
		в/из TWDR	в регистр TWCR				
			STA	STO	TWINT	TWEA	
$98	Устройство уже адресовано (общий вызов): был принят байт данных и послано неподтверждение (NACK)	Считать данные	0	0	1	0	Переключиться в режим неадресованного ведомого; распознавание любых вызовов отключено
		Считать данные	0	0	1	1	Переключиться в режим неадресованного ведомого; разрешено распознавание SLA с собственным адресом; разрешено распознавание общих вызовов, если TWGCE = «1»
		Считать данные	1	0	1	0	Переключиться в режим неадресованного ведомого; распознавание любых вызовов запрещено; после освобождения шины будет сформировано состояние СТАРТ
		Считать данные	1	0	1	1	Переключиться в режим неадресованного ведомого; разрешено распознавание SLA с собственным адресом; разрешено распознавание общих вызовов, если TWGCE = «1»; после освобождения шины будет сформировано состояние СТАРТ
$A0	Было обнаружено состояние СТАРТ или ПОВСТАРТ в то время, когда устройство было адресовано в качестве ведомого	Считать данные	0	0	1	0	Переключиться в режим неадресованного ведомого; распознавание любых вызовов отключено
		Считать данные	0	0	1	1	Переключиться в режим неадресованного ведомого; разрешено распознавание SLA с собственным адресом; разрешено распознавание общих вызовов, если TWGCE = «1»
		Считать данные	1	0	1	0	Переключиться в режим неадресованного ведомого; распознавание любых вызовов запрещено; после освобождения шины будет сформировано состояние СТАРТ
		Считать данные	1	0	1	1	Переключиться в режим неадресованного ведомого; разрешено распознавание SLA с собственным адресом; разрешено распознавание общих вызовов, если TWGCE = «1»; после освобождения шины будет сформировано состояние СТАРТ

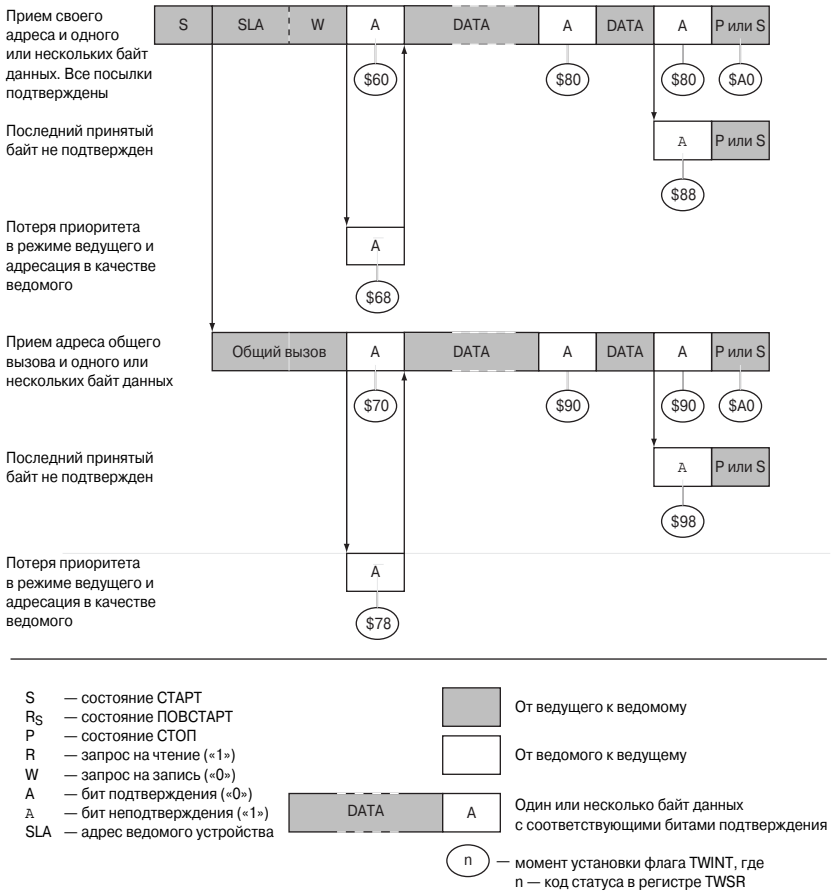
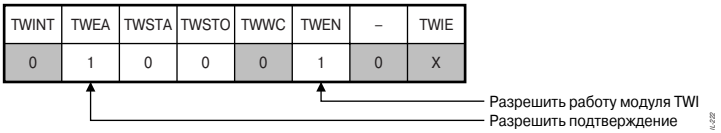


Рис. 2.125. Состояния модуля TWI в режиме «Ведомый приемник»

18.5.4. Режим «Ведомый передатчик»

В режиме «Ведомый передатчик» ведомое устройство осуществляет передачу данных ведущему, который в этом случае является приемником. Перед тем как переключить модуль в этот режим, следует занести в старшие разряды регистра TWAR адрес устройства и в соответствии с логикой работы программы установить или сбросить младший разряд регистра (TWGCE). Затем необходимо записать в регистр TWCR следующее значение:



После инициализации регистров TWAR и TWCR модуль будет ожидать адресного пакета с адресом, указанным в регистре TWAR, или общего вызова (если его распознавание разрешено). Значение следующего за адресом бита направления определит режим, в который переключится модуль. Если этот бит будет установлен в «1» (чтение), модуль TWI переключится в режим «Ведомый передатчик». В противном случае модуль переключится в режим «Ведомый приемник». Следует помнить, что устройство также может автоматически переключиться в режим «Ведомый приемник» из режима ведущего при потере приоритета (см. описание кода статуса $B0 в **Табл. 2.148**).

После приема пакета SLA+R установится флаг TWINT, и состояние обмена можно будет, как обычно, определить по коду статуса. Возможные действия со стороны программы, соответствующие тому или иному коду статуса, указаны в **Табл. 2.148**.

Таблица 2.148. **Коды статуса для режима «Ведомый-передатчик»**

Код статуса	Состояние шины и модуля TWI	Действия программы					Следующее действие, выполняемое модулем TWI
		в/из TWDR	в регистр TWCR				
			STA	STO	TWINT	TWEA	
$A8	Был принят SLA+W с собственным адресом и послано подтверждение (ACK)	Загрузить данные	X	0	1	0	Будет передан последний байт данных; должно быть получено неподтверждение (NACK)
		Загрузить данные	X	0	1	1	Будет передан очередной байт данных; должно быть получено подтверждение (ACK)
$B0	Потеря приоритета в режиме ведущего во время передачи SLA+R/W; был принят SLA+W с собственным адресом и послано подтверждение (ACK)	Загрузить данные	X	0	1	0	Будет передан последний байт данных; должно быть получено неподтверждение (NACK)
		Загрузить данные	X	0	1	1	Будет передан очередной байт данных; должно быть получено подтверждение (ACK)
$B8	Был передан байт данных и получено подтверждение (ACK)	Загрузить данные	X	0	1	0	Будет передан последний байт данных; должно быть получено неподтверждение (NACK)
		Загрузить данные	X	0	1	1	Будет передан очередной байт данных; должно быть получено подтверждение (ACK)

Продолжение таблицы 2.148

Код статуса	Состояние шины и модуля TWI	Действия программы					Следующее действие, выполняемое модулем TWI
		в/из TWDR	в регистр TWCR				
			STA	STO	TWINT	TWEA	
$C0	Был передан байт данных и получено неподтверждение (NACK)	Нет действий	0	0	1	0	Переключиться в режим неадресованного ведомого; распознавание любых вызовов запрещено
		Нет действий	0	0	1	1	Переключиться в режим неадресованного ведомого; разрешено распознавание SLA с собственным адресом; разрешено распознавание общих вызовов, если TWGCE = «1»
		Нет действий	1	0	1	0	Переключиться в режим неадресованного ведомого; распознавание любых вызовов запрещено; после освобождения шины будет сформировано состояние СТАРТ
		Нет действий	1	0	1	1	Переключиться в режим неадресованного ведомого; разрешено распознавание SLA с собственным адресом; разрешено распознавание общих вызовов, если TWGCE = «1»; после освобождения шины будет сформировано состояние СТАРТ
$C8	Был передан последний байт данных и получено подтверждение (ACK)	Нет действий	0	0	1	0	Переключиться в режим неадресованного ведомого; распознавание любых вызовов запрещено
		Нет действий	0	0	1	1	Переключиться в режим неадресованного ведомого; разрешено распознавание SLA с собственным адресом; разрешено распознавание общих вызовов, если TWGCE = «1»
		Нет действий	1	0	1	0	Переключиться в режим неадресованного ведомого; распознавание любых вызовов запрещено; после освобождения шины будет сформировано состояние СТАРТ
		Нет действий	1	0	1	1	Переключиться в режим неадресованного ведомого; разрешено распознавание SLA с собственным адресом; разрешено распознавание общих вызовов, если TWGCE = «1»; после освобождения шины будет сформировано состояние СТАРТ

При передаче последнего байта данных необходимо сбросить в «0» разряд TWEA. После этого модуль перейдет в состояние с кодом статуса $C0 или $C8 в зависимости от того, какой сигнал (ACK или NACK) передаст ведущий в ответ. В состояние с кодом статуса $C8 модуль TWI перейдет в случае, если ведущий затребовал дополнительные данные, передав

подтверждение (ACK), несмотря на то что ведомый передатчик передал последний байт и ожидал сигнала NACK.

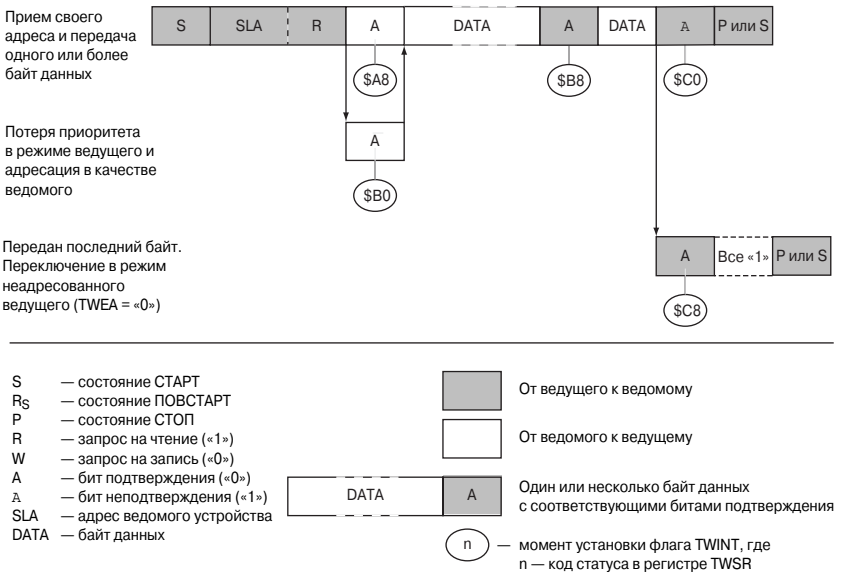


Рис. 2.126. Состояния модуля TWI в режиме «Ведомый передатчик»

После перехода в любое из указанных состояний модуль TWI будет игнорировать обращения к нему ведущего. Соответственно, если ведущий будет продолжать обмен по шине, он будет постоянно принимать значение «1». Обработка адресных пакетов при сброшенном разряде TWEA также прекращается, но может быть возобновлена в любой момент времени повторной установкой этого разряда.

18.5.5. Комбинирование различных режимов

На практике для выполнения какой-либо операции по шине TWI каждому устройству приходится использовать несколько режимов, переключаясь между ними при необходимости. В качестве примера рассмотрим операцию чтения данных из внешнего EEPROM.

Все операции такого рода можно разбить на четыре этапа:

1) инициирование обмена;
2) передача адреса, по которому требуется прочитать данные;
3) выполнение чтения;
4) завершение передачи.

Из сказанного видно, что при выполнении операции происходит передача информации как от ведущего к ведомому, так и наоборот. После инициирования обмена по шине ведущий должен находиться в режиме «Ведущий передатчик», чтобы сообщить ведомому адрес, по которому он намеревается осуществить чтение. Для выполнения следующего этапа операции (чтение данных) ведущий должен переключиться в режим «Ведущий приемник». При этом он должен сохранять контроль над шиной во время выполнения всех этапов операции. Для этого используется состояние ПОВСТАРТ. В рассматриваемом случае ведущий формирует это состояние между передачей адреса и приемом данных, как показано на **Рис. 2.127**.

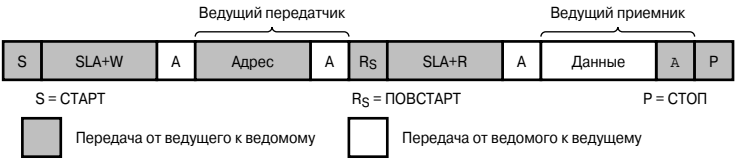


Рис. 2.127. Пример обращения к внешнему EEPROM

18.5.6. Арбитраж

В случае присутствия на шине нескольких ведущих возможна ситуация, при которой несколько ведущих одновременно начнут процесс обмена. Для таких случаев спецификацией TWI предусмотрен процесс распределения приоритетов (арбитраж), в результате выполнения которого на шине остается только одно ведущее устройство. Принципы этого процесса были рассмотрены в параграфе 2.2, однако его выполнение может развиваться по различным сценариям (**Рис. 2.128**):

1. Два или более ведущих осуществляют однотипный обмен с одним и тем же ведомым. В этом случае никто из них не сможет распознать конфликт на шине.
2. Два или более ведущих обращаются к одному и тому же ведомому с различными данными или различными типами обмена (чтение/запись). В этом случае распределение приоритетов произойдет во время передачи битов данных или бита направления. Ведущий, потерявший приоритет, может либо переключиться в режим неадресованного ведомого, либо дождаться освобождения шины и сформировать состояние СТАРТ для ее захвата.
3. Два или более ведущих обращаются к различным ведомым. В этом случае распределение приоритетов начнется во время передачи битов адресного пакета. Ведущий, потерявший приоритет, переключится в режим ведомого для проверки, не

был ли он адресован ведущим. Если бывший ведущий был адресован, он переключится в режим «Ведомый передатчик» или «Ведомый приемник» в зависимости от значения бита направления. Если же он не был адресован, он переключится в режим неадресованного ведомого, либо дождется освобождения шины и сформирует новое состояние СТАРТ.

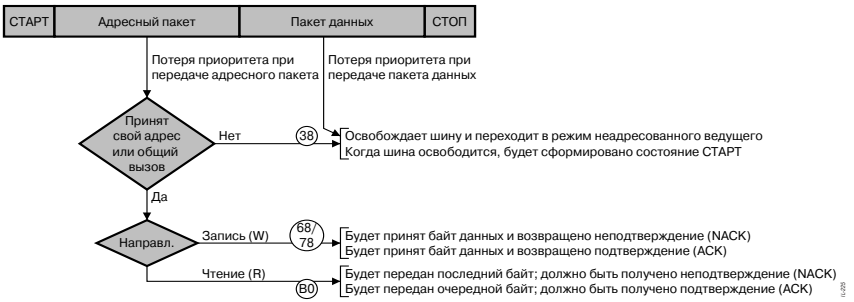


Рис. 2.128. Состояния модуля TWI во время арбитража

18.6. Параметры интерфейса TWI

Требования, предъявляемые к устройствам, которые подключаются к шине TWI, приведены в **Табл. 2.149**. Разумеется, модуль TWI микроконтроллеров семейства Mega полностью им удовлетворяет. Временные диаграммы сигналов, формируемых на шине TWI, приведены на **Рис. 2.129**.

Таблица 2.149. **Требования шины TWI**

Параметр		Условия	min	max
V_{IL}	Входное напряжение НИЗКОГО уровня [В]	—	-0.5	$0.3V_{CC}$
V_{IH}	Входное напряжение ВЫСОКОГО уровня [В]	—	$0.7V_{CC}$	$V_{CC} + 0.5$
V_{HYS}	Гистерезис входных каскадов [В]	—	$0.05V_{CC}$	—
V_{OL}	Выходное напряжение НИЗКОГО уровня [В]	—	0	0.4
t_r	Время нарастания сигнала [нс]	—	$20 + 0.1C_b$	300
t_{OF}	Длительность спада сигнала с V_{IHmin} до V_{ILmax} [нс]	10 пФ < C_b < 400 пФ (C_b – емкость одной линии)	$20 + 0.1C_b$	250
t_{SP}	Длительность импульсов, подавляемых входным фильтром [нс]	—	0	50
I_i	Входной ток по каждому выводу [мкА]	$0.1V_{CC} < V_i < 0.9V_{CC}$	–10	10

Продолжение таблицы 2.149

Параметр		Условия		min	max
C_i	Емкость каждого вывода [пФ]	—		—	10
f_{SCL}	Тактовая частота шины [кГц]	$f_{CK} > \max\{16f_{SCL}; 250\text{ кГц}\}$		0	400
R_p	Сопротивление подтягивающего резистора [Ом]	f_{SCL} [кГц]	≤ 100	$\frac{V_{CC} - 0.4\text{ В}}{3\text{ мА}}$	$\frac{1000\text{ нс}}{3\text{ мА}}$
			> 100	$\frac{V_{CC} - 0.4\text{ В}}{3\text{ мА}}$	$\frac{300\text{ нс}}{3\text{ мА}}$
$t_{HD;STA}$	Время удержания состояния СТАРТ [мкс]		≤ 100	4.0	—
			> 100	0.6	—
t_{LOW}	Длительность паузы между тактовыми импульсами [мкс]		≤ 100	4.7*	—
			> 100	1.3*	—
t_{HIGH}	Длительность тактовых импульсов [мкс]		≤ 100	4.0	—
			> 100	0.6	—
$t_{SU;STA}$	Задержка формирования состояния ПОВСТАРТ [мкс]		≤ 100	4.7	—
			> 100	0.6	—
$t_{HD;DAT}$	Время удержания данных [мкс]		≤ 100	0	3.45
			> 100	0	0.9
$t_{SU;DAT}$	Задержка выдачи данных [нс]		≤ 100	250	—
			> 100	100	—
$t_{SU;SAO}$	Задержка формирования состояния СТОП [мкс]		≤ 100	4.0	—
			> 100	0.6	—
t_{BUF}	Длительность нахождения шины в незанятом состоянии между состояниями СТОП и СТАРТ [мкс]		≤ 100	4.7	—
			> 100	1.3	—

* Действительная величина паузы между тактовыми импульсами, формируемыми модулем TWI, равна ($1/f_{SCL} - 2/f_{CK}$).

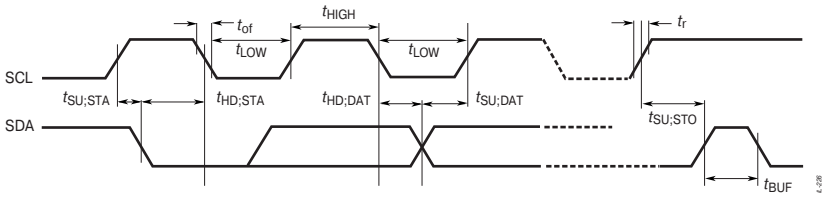


Рис. 2.129. Временные диаграммы сигналов шины TWI

Часть 3
Команды микроконтроллеров семейств Tiny и Mega

Глава 19. **Общие сведения о системе команд**

19.1. Введение в систему команд

Система команд микроконтроллеров AVR семейств Tiny и Mega весьма развита и насчитывает в различных моделях от 90 до 133 различных инструкций. Несмотря на то что микроконтроллеры AVR являются микроконтроллерами с RISC-архитектурой (процессор с сокращенным набором команд), по количеству реализованных инструкций и их разнообразию они больше похожи на микроконтроллеры с CISC-архитектурой (процессор с полным набором команд). Практически каждая из команд (за исключением команд, у которых одним из операндов является 16-разрядный адрес) занимает только одну ячейку памяти программ. Причем это достигнуто не за счет сокращения количества команд процессора, а за счет увеличения разрядности памяти программ.

19.2. Операнды

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

Для обращения к различным областям адресного пространства памяти данных используются различные команды, реализующие, в свою очередь, различные способы адресации. Подробно способы адресации памяти данных для моделей семейства Tiny были рассмотрены в Главе 2, а для моделей семейства Mega — в Главе 9.

Доступ к регистрам ввода/вывода осуществляется по их адресам, являющимися операндами команды. Вместе с тем при написании ассемблерных программ гораздо удобнее обращаться к регистрам, используя вместо числовых значений адресов их стандартные, принятые

в фирменной документации, символические имена. Чтобы задать соответствие этих имен реальным адресам необходимо подключить в начале программы (при помощи директивы ассемблера .INCLUDE) файл определения адресов регистров ввода/вывода. Помимо всего прочего, такое решение облегчит перенос программного обеспечения с одного типа кристалла на другой.

Эти файлы (для каждой модели микроконтроллеров семейства) свободно распространяются фирмой «Atmel» вместе с документацией на микроконтроллеры (в частности, они находятся на web-сайте фирмы). Для РОН, входящих в состав индексных регистров, в этих файлах определяются также дополнительные символические имена (**Табл. 3.1**).

Таблица 3.1. **Дополнительные символические имена индексных регистров**

Регистр	Символическое имя
R26	XL
R27	XH
R28	YL
R29	YH
R30	ZL
R31	ZH

Названия этих файлов унифицированы и определяются следующим образом:

```
<номер_модели>def.inc
```

Например, программа для микроконтроллера ATtiny15L должна содержать следующую директиву ассемблера:

```
.include "tn15def.inc"
```

а для микроконтроллера ATmega128x:

```
.include "m128def.inc"
```

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

Как уже было упомянуто, в микроконтроллерах семейства Tiny и Mega память программ является 16-разрядной. Соответственно большинство команд описываются 16-разрядным словом, которое называется также кодом операции (КОП). Код операции — это число, расположенное в памяти программ и определяющее действие, которое необходимо произвести между источником и приемником. Некоторые команды, у которых один из операндов является 16-разрядным адресом, занимают две ячейки памяти программ. Соответственно, код операции таких команд является 4-байтным числом.

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

Некоторые константы, которые могут быть полезны при написании программ, определены в упомянутых включаемых файлах:

RAMEND — значение верхнего адреса внутреннего ОЗУ (для моделей семейства Tiny, рассматриваемых в книге, эта константа не определена);

XRAMEND — значение верхнего адреса внешнего ОЗУ (для моделей, не поддерживающих подключение внешнего ОЗУ, эта константа не определена);

E2END — значение верхнего адреса EEPROM (для моделей, не имеющих в своем составе EEPROM-памяти, эта константа не определена);

FLASHEND — значение верхнего адреса памяти программ.

Для микроконтроллеров семейства Mega, кроме того, определен ряд дополнительных констант, связанных с такой возможностью этих микроконтроллеров, как самопрограммирование (см. Главу 26):

SMALLBOOTSTART — наименьший размер области загрузчика;

SECONDBOOTSTART — вторая возможная величина области загрузчика;

THIRDBOOTSTART — третья возможная величина области загрузчика;

LARGEBOOTSTART — наибольший размер области загрузчика;

PAGESIZE — размер страницы памяти программ в 2-байтовых словах.

19.3. Типы команд

Все множество команд микроконтроллеров AVR семейств Tiny и Mega можно разбить на несколько групп:

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

Далее подробно описана каждая группа команд.

19.3.1. Команды логических операций

Команды логических операций позволяют выполнять стандартные логические операции над байтами, такие, как логическое умножение (И), логическое сложение (ИЛИ), операцию «исключающее

ИЛИ», а также вычисление обратного (дополнение до единицы) и дополнительного (дополнение до двух) кодов числа. К этой группе можно отнести также команды очистки/установки регистров и команду перестановки тетрад. Операции производятся между регистрами общего назначения либо между регистром и константой; результат сохраняется в РОН. Все команды из этой группы выполняются за один машинный цикл.

19.3.2. Команды арифметических операций и команды сдвига

К данной группе относятся команды, позволяющие выполнять такие базовые операции, как сложение, вычитание, сдвиг (вправо и влево), инкремент и декремент. В микроконтроллерах семейства Mega также имеются команды, позволяющие осуществлять умножение 8-разрядных значений. Все операции производятся только над регистрами общего назначения. При этом микроконтроллеры AVR позволяют легко оперировать как знаковыми, так и беззнаковыми числами, а также работать с числами, представленными в дополнительном коде.

Почти все команды рассматриваемой группы выполняются за один машинный цикл. Команды умножения и команды, оперирующие двухбайтовыми значениями, выполняются за два цикла.

19.3.3. Команды операций с битами

К данной группе относятся команды, выполняющие установку или сброс заданного разряда РОН или РВВ. Причем для изменения разрядов регистра состояния SREG имеются также дополнительные команды (точнее говоря, эквивалентные мнемонические обозначения общих команд), т. к. проверка состояния разрядов именно этого регистра производится чаще всего. Условно к этой группе можно отнести также две команды передачи управления типа «проверка/пропуск», которые пропускают следующую команду в зависимости от состояния разряда РОН или РВВ.

Все задействованные разряды РВВ имеют свои символические имена. Определения этих имен описаны в том же включаемом файле, что и определения символических имен адресов регистров (см. раздел 19.2). Таким образом, после включения в программу указанного файла в командах вместо числовых значений номеров разрядов можно будет указывать их символические имена.

Следует помнить, что в командах CBR и SBR операндом является битовая маска, а не номер разряда. Для получения битовой маски из номера разряда следует воспользоваться ассемблерным оператором

«сдвиг влево» («<<»), как показано в следующем примере:

```
sbr r16,(1<<SE)+(1<<SM)
out MCUCR,r16      ;Установить флаги SE и SM регистра MCUCR
```

Всем командам данной группы требуется один машинный цикл для выполнения, за исключением случаев, когда в результате проверки происходит пропуск команды. В этом случае команда выполняется за 2 или 3 машинных цикла в зависимости от пропускаемой команды.

19.3.4. Команды пересылки данных

Команды этой группы предназначены для пересылки содержимого ячеек, находящихся в адресном пространстве памяти данных. Разделение адресного пространства на три части (РОН, РВВ, ОЗУ) предопределило разнообразие команд данной группы. Пересылка данных, выполняемая командами группы, может производиться в следующих направлениях:

- РОН ⇔ РОН;
- РОН ⇔ РВВ;
- РОН ⇔ память данных.

Также к данной группе можно отнести стековые команды PUSH и POP (отсутствуют в микроконтроллерах семейства Tiny), позволяющие сохранять в стеке и восстанавливать из стека содержимое РОН.

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

19.3.5. Команды передачи управления

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

В системе команд микроконтроллеров семейства имеются команды как безусловного, так и условного переходов. Команды относительного перехода (`RJMP`), а в микроконтроллерах семейства Mega также косвенного (`IJMP`) и абсолютного (`JMP`) безусловного перехода являются самыми простыми в этой группе. Их функция заключается только в записи нового адреса в счетчик команд. Команды условного перехода также изменяют содержимое счетчика команд, однако это изменение происходит только при выполнении

некоторого условия или, точнее, при определенном состоянии различных флагов регистра SREG.

Все команды условного перехода можно разбить на две подгруппы. Первая подгруппа — команды условного перехода общего назначения. В эту подгруппу входят две команды `BRBS s,k` и `BRBC s,k`, в которых явно задается номер тестируемого флага регистра SREG. Соответственно, переход осуществляется при SREG.s = 0 (`BRBC`) или SREG.s = 1 (`BRBS`). Другую подгруппу составляют 18 специализированных команд, каждая из которых выполняет переход по какому-либо конкретному условию («равно», «больше или равно», «был перенос» и т. п.). Причем одни команды используются после сравнения беззнаковых чисел, другие — после сравнения чисел со знаком. Возможные проверяемые условия, а также соответствующие им команды условного перехода приведены в **Табл. 3.2**.

Таблица 3.2. **Сводная таблица команд условного перехода**

Проверка	Логическое условие	Команда	Обратная проверка	Логическое условие	Команда	Тип данных
Rd > Rr	Z • (N ⊕ V) = 0	`BRLT*`	Rd ≤ Rr	Z + (N ⊕ V) = 1	`BRGE*`	Со знаком
Rd ≥ Rr	(N ⊕ V) = 0	`BRGE`	Rd < Rr	(N ⊕ V) = 1	`BRLT`	Со знаком
Rd = Rr	Z = 1	`BREQ`	Rd ≠ Rr	Z = 0	`BRNE`	Со знаком
Rd ≤ Rr	Z + (N ⊕ V) = 1	`BRGE(*)`	Rd > Rr	Z • (N ⊕ V) = 0	`BRLT*`	Со знаком
Rd < Rr	(N ⊕ V) = 1	`BRLT`	Rd ≥ Rr	(N ⊕ V) = 0	`BRGE`	Со знаком
Rd > Rr	C + Z = 0	`BRLO(*)`	Rd ≤ Rr	C + Z = 1	`BRSH*`	Без знака
Rd ≥ Rr	C = 0	`BRHS/ BRCC`	Rd < Rr	C = 1	`BRLO/ BRCS`	Без знака
Rd = Rr	Z = 1	`BREQ`	Rd ≠ Rr	Z = 0	`BRNE`	Без знака
Rd ≤ Rr	C = Z = 1	`BRSH (*)`	Rd > Rr	C = Z = 0	`BRLO*`	Без знака
Rd < Rr	C = 1	`BRLO/ BRCS`	Rd ≥ Rr	C = 0	`BRSH/ BRCC`	Без знака
«Перенос»	C = 1	`BRCS`	«Нет переноса»	C = 0	`BRCC`	—
«Меньше нуля»	N = 1	`BRMI`	«Больше нуля»	N = 0	`BRPL`	—

Продолжение таблицы 3.2

Проверка	Логическое условие	Команда	Обратная проверка	Логическое условие	Команда	Тип данных
«Переполнение»	V = 1	BRVS	«Нет переполнения»	V = 0	BRVC	—
«Ноль»	Z = 1	BREQ	«Не ноль»	Z = 0	BRNE	—

* Для перехода по этому условию операнды предшествующей команды сравнения должны быть записаны в обратном порядке, т. е. вместо `CP Rd,Rr` → `CP Rr,Rd`.

Вообще говоря, команды, указанные в **Табл. 3.2**, являются всего лишь эквивалентными мнемоническими обозначениями команд `BRBS s,k` и `BRBC s,k` с определенными значениями операнда «s». Команда `BREQ k` имеет, например, такой же код операции, что и команда `BRBS 1,k`, а команда `BRGE k` — `BRBC 4,k`.

Команды вызова подпрограммы (`ICALL`, `RCALL` и `CALL`) работают практически так же, как и команды безусловного перехода. Отличие заключается в том, что, перед тем как выполнить переход, значение счетчика команд сохраняется в стеке. Кроме того, подпрограмма должна заканчиваться командой возврата `RET`, как показано в следующем примере:

```
...
rcall sp_test       ;Вызов подпрограммы «sp_test»
...                 ;Текст основной программы
sp_test             ;Метка подпрограммы
push r2             ;Сохранить r2 в стеке
...                 ;Выполнение подпрограммы
pop r2              ;Восстановить r2 из стека
ret                 ;Возврат из подпрограммы
```

В приведенном примере команда `RET` заменяет адрес, находящийся в счетчике команд, адресом команды, следующей за командой `CALL`.

Очевидно, что команды передачи управления нарушают нормальное (линейное) выполнение основной программы. Каждый раз, когда выполняется команда из этой группы (кроме команд сравнения), нормальное функционирование конвейера нарушается. Перед загрузкой в конвейер нового адреса производится остановка и очистка выполняемой последовательности команд. Соответственно, реинициализация конвейера приводит к необходимости использования нескольких машинных циклов для выполнения таких команд. Чтобы получить более точную информацию, обратитесь к таблицам, приведенным в разделе 19.4.

19.3.6. Команды управления системой

В эту группу входят всего 3 команды:

- `NOP` — пустая команда;
- `SLEEP` — перевод микроконтроллера в режим пониженного энергопотребления;
- `WDR` — сброс сторожевого таймера.

Все команды этой группы выполняются за один машинный цикл.

19.4. Сводные таблицы команд

В **Табл. 3.3...3.8** указаны все команды, которыми располагают микроконтроллеры AVR семейств Tiny и Mega. В каждой таблице команды сгруппированы по функциональному признаку. В таблицах приведены такие основные сведения о командах, как мнемоническое обозначение команды, ее описание, число машинных циклов, необходимых для ее выполнения, а также флаги регистра SREG, на которые воздействует эта команда. Информация в таблицах изложена в сжатом виде, а детальное описание всех команд приведено во 2-й главе. Команды, не поддерживаемые микроконтроллерами семейства Tiny, выделены в таблицах серым цветом.

Таблица 3.3. **Группа команд логических операций**

Мнемоника	Описание	Операция	Циклы	Флаги
`AND Rd,Rr`	«Логическое И» двух РОН	Rd = Rd • Rr	1	Z,N,V
`ANDI Rd,K`	«Логическое И» РОН и константы	Rd = Rd • K	1	Z,N,V
`EOR Rd,Rr`	«Исключающее ИЛИ» двух РОН	Rd = Rd ⊕ Rr	1	Z,N,V
`OR Rd,Rr`	«Логическое ИЛИ» двух РОН	Rd = Rd ∨ Rr	1	Z,N,V
`ORI Rd,K`	«Логическое ИЛИ» РОН и константы	Rd = Rd ∨ K	1	Z,N,V
`COM Rd`	Перевод в обратный код	Rd = $FF – Rd	1	Z,C,N,V
`NEG Rd`	Перевод в дополнительный код	Rd = $00 – Rd	1	Z,C,N,V,H
`CLR Rd`	Сброс всех разрядов РОН	Rd = Rd ⊕ Rd	1	Z,N,V
`SER Rd`	Установка всех разрядов РОН	Rd = $FF	1	–
`TST Rd`	Проверка РОН на отрицательное или нулевое значение	Rd • Rd	1	Z,N,V
`SWAP Rd`	Обмен местами тетрад в РОН	Rd(3...0) = Rd(7...4), Rd(7...4) = Rd(3...0)	1	–

Таблица 3.4. **Группа команд арифметических операций**

Мнемоника	Описание	Операция	Циклы	Флаги
ADD Rd,Rr	Сложение двух РОН	Rd = Rd + Rr	1	Z,C,N,V,H
ADC Rd,Rr	Сложение двух РОН с переносом	Rd = Rd + Rr + C	1	Z,C,N,V,H
ADIW Rd,K	Сложение регистровой пары с константой	Rdh:Rdl = Rdh:Rdl + K	2	Z,C,N,V,S
SUB Rd,Rr	Вычитание двух РОН	Rd = Rd – Rr	1	Z,C,N,V,H
SUBI Rd,K	Вычитание константы из РОН	Rd = Rd – K	1	Z,C,N,V,H
SBC Rd,Rr	Вычитание двух РОН с заемом	Rd = Rd – Rr – C	1	Z,C,N,V,H
SBCI Rd,K	Вычитание константы из РОН с заемом	Rd = Rd – K – C	1	Z,C,N,V,H
SBIW Rd,K	Вычитание константы из регистровой пары	Rdh:Rdl = Rdh:Rdl – K	2	Z,C,N,V,S
DEC Rd	Декремент РОН	Rd = Rd – 1	1	Z,N,V
INC Rd	Инкремент РОН	Rd = Rd + 1	1	Z,N,V
ASR Rd	Арифметический сдвиг вправо	Rd(n) = Rd(n + 1), n = 0...6	1	Z,C,N,V
LSL Rd	Логический сдвиг влево	Rd(n + 1) = Rd(n), Rd(0) = 0	1	Z,C,N,V
LSR Rd	Логический сдвиг вправо	Rd(n) = Rd(n + 1), Rd(7) = 0	1	Z,C,N,V
ROL Rd	Сдвиг влево через перенос	Rd(0) = C, Rd(n + 1) = Rd(n), C = Rd(7)	1	Z,C,N,V
ROR Rd	Сдвиг вправо через перенос	Rd(7) = C, Rd(n) = Rd(n + 1), C = Rd(0)	1	Z,C,N,V
MUL Rd,Rr	Умножение беззнаковых чисел	R1:R0 = Rd × Rr	2	Z,C
MULS Rd,Rr	Умножение чисел со знаком	R1:R0 = Rd × Rr	2	Z,C
MULSU Rd,Rr	Умножение беззнакового числа на число со знаком	R1:R0 = Rd × Rr	2	Z,C
FMUL Rd,Rr	Умножение дробных беззнаковых чисел	R1:R0 = (Rd × Rr) << 1	2	Z,C
FMULS Rd,Rr	Умножение дробных чисел со знаком	R1:R0 = (Rd × Rr) << 1	2	Z,C
FMULSU Rd,Rr	Умножение дробного беззнакового числа и дробного числа со знаком	R1:R0 = (Rd × Rr) << 1	2	Z,C

Таблица 3.5. **Группа команд операций с битами**

Мнемоника	Описание	Операция	Циклы	Флаги
CBR Rd,K	Сброс разряда(ов) РОН	Rd = Rd • ($FF – K)	1	Z, N,V
SBR Rd,K	Установка разряда(ов) РОН	Rd = Rd ∨ K	1	Z, N,V
CBI A,b	Сброс разряда РВВ	A.b = 0	2	—
SBI A,b	Установка разряда РВВ	A.b = 1	2	—
BCLR s	Сброс флага	SREG.s = 0	1	SREG.s
BSET s	Установка флага	SREG.s = 1	1	SREG.s
BLD Rd,b	Загрузка разряда РОН из флага T (SREG)	Rd.b = T	1	—
BST Rr,b	Запись разряда РОН в флаг T (SREG)	T = Rr.b	1	T
CLC	Сброс флага переноса	C = 0	1	C
SEC	Установка флага переноса	C = 1	1	C
CLN	Сброс флага отр. числа	N = 0	1	N
SEN	Установка флага отр. числа	N = 1	1	N
CLZ	Сброс флага нуля	Z = 0	1	Z
SEZ	Установка флага нуля	Z = 1	1	Z
CLI	Общее запрещение прерываний	I = 0	1	I
SEI	Общее разрешение прерываний	I = 1	1	I
CLS	Сброс флага знака	S = 0	1	S
SES	Установка флага знака	S = 1	1	S
CLV	Сброс флага переполнения доп. кода	V = 0	1	V
SEV	Установка флага переполнения доп. кода	V = 1	1	V
CLT	Сброс флага T	T = 0	1	T
SET	Установка флага T	T = 1	1	T
CLH	Сброс флага половинного переноса	H = 0	1	H
SEH	Установка флага половинного переноса	H = 1	1	H

Таблица 3.6. **Группа команд пересылки данных**

Мнемоника	Описание	Операция	Циклы	Флаги
MOV Rd,Rr	Пересылка между РОН	Rd = Rr	1	—
MOVW Rd,Rr	Пересылка двухбайтовых значений	Rd + 1:Rd = Rr + 1:Rr	1	—
LDI Rd,K	Загрузка константы в РОН	Rd = K	1	—
LD Rd,X	Косвенное чтение	Rd = [X]	2	—
LD Rd,X+	Косвенное чтение с постинкрементом	Rd = [X], X = X + 1	2	—
LD Rd,-X	Косвенное чтение с преддекрементом	X = X – 1, Rd = [X]	2	—

Продолжение таблицы 3.6

Мнемоника	Описание	Операция	Циклы	Флаги
LD Rd,Y	Косвенное чтение	Rd = [Y]	2	—
LD Rd,Y+	Косвенное чтение с постинкрементом	Rd = [Y], Y = Y + 1	2	—
LD Rd,-Y	Косвенное чтение с преддекрементом	Y = Y – 1, Rd = [Y]	2	—
LDD Rd,Y+q	Косвенное относительное чтение	Rd = [Y+q]	2	—
LD Rd,Z	Косвенное чтение	Rd = [Z]	2	—
LD Rd,Z+	Косвенное чтение с постинкрементом	Rd = [Z], Z = Z + 1	2	—
LD Rd,-Z	Косвенное чтение с преддекрементом	Z = Z – 1, Rd = [Z]	2	—
LDD Rd,Z+q	Косвенное относительное чтение	Rd = [Z + q]	2	—
LDS Rd,k	Непосредственное чтение из ОЗУ	Rd = [k]	2	—
ST X,Rr	Косвенная запись	[X] = Rr	2	—
ST X+,Rr	Косвенная запись с постинкрементом	[X] = Rr, X = X + 1	2	—
ST -X,Rr	Косвенная запись с преддекрементом	X = X – 1, [X] = Rr	2	—
ST Y,Rr	Косвенная запись	[Y] = Rr	2	—
ST Y+,Rr	Косвенная запись с постинкрементом	[Y] = Rr, Y = Y + 1	2	—
ST -Y,Rr	Косвенная запись с преддекрементом	Y = Y – 1, [X] = Rr	2	—
STD Y+q,Rr	Косвенная относительная запись	[Y+q] = Rr	2	—
ST Z,Rr	Косвенная запись	[Z] = Rr	2	—
ST Z+,Rr	Косвенная запись с постинкрементом	[Z] = Rr, Z = Z + 1	2	—
ST -Z,Rr	Косвенная запись с преддекрементом	Z = Z – 1, [Z] = Rr	2	—
STD Z+q,Rr	Косвенная относительная запись	[Z + q] = Rr	2	—
STS k,Rr	Непосредственная запись в ОЗУ	[k] = Rr	2	—
LPM	Загрузка данных из памяти программ	R0 = {Z}	3	—
LPM Rd,Z	Загрузка данных из памяти программ	Rв = {Z}	3	—
LPM Rd,Z+	Загрузка данных из памяти программ с постинкрементом	Rв = {Z}, Z = Z + 1	3	—
ELPM	Расширенная загрузка данных из памяти программ	R0 = {RAMPZ:Z}	3	—
ELPM Rd,Z	Расширенная загрузка данных из памяти программ	Rв = {RAMPZ:Z}	3	—
ELPM Rd,Z+	Расширенная загрузка данных из памяти программ с постинкрементом	Rв = {RAMPZ:Z}, RAMPZ:Z = RAMPZ:Z + 1	3	—
SPM	Запись в память программ	{Z} = R1:R0	—	—
IN Rd,A	Пересылка из РВВ в РОН	Rd = A	1	—
OUT A,Rr	Пересылка из РОН в РВВ	A = Rr	1	—
PUSH Rr	Сохранение байта в стеке	STACK = Rr	2	—
POP Rd	Извлечение байта из стека	Rd = STACK	2	—

Таблица 3.7. **Группа команд передачи управления**

Мнемоника	Описание	Операция	Циклы	Флаги
RJMP k	Относительный безусловный переход	PC = PC + k + 1	2	—
IJMP	Косвенный безусловный переход	PC = Z	2	—
JMP k	Абсолютный переход	PC = k	3	—
RCALL k	Относительный вызов подпрограммы	PC = PC + k + 1	3	—
ICALL	Косвенный вызов подпрограммы	PC = Z	3	—
CALL k	Абсолютный вызов подпрограммы	PC = k	4	—
RET	Возврат из подпрограммы	PC = STACK	4	—
RETI	Возврат из подпрограммы обработки прерывания	PC = STACK	4	I
CP Rd,Rr	Сравнение РОН	Rd – Rr	1	Z, N, V, C, H
CPC Rd,Rr	Сравнение РОН с учетом переноса	Rd – Rr – C	1	Z, N, V, C, H
CPI Rd,K	Сравнение РОН с константой	Rd – K	1	Z, N, V, C, H
CPSE Rd,Rr	Сравнение и пропуск следующей команды при равенстве	Если Rd = Rr, то PC = PC + 2 (3)	1/2/3	—
SBRC Rr,b	Пропуск след. команды, если разряд РОН сброшен	Если Rr.b = 0, то PC = PC + 2 (3)	1/2/3	—
SBRS Rr,b	Пропуск след. команды, если разряд РОН установлен	Если Rr.b = 1, то PC = PC + 2 (3)	1/2/3	—
SBIC A,b	Пропуск след. команды, если разряд РВВ сброшен	Если A.b = 0, то PC = PC + 2 (3)	1/2/3	—
SBIS A,b	Пропуск след. команды, если разряд РВВ установлен	Если A.b = 1, то PC = PC + 2 (3)	1/2/3	—
BRBC s,k	Переход, если флаг s регистра SREG сброшен	Если SREG.s = 0, то PC = PC + k + 1	1/2	—
BRBS s,k	Переход, если флаг s регистра SREG установлен	Если SREG.s = 1, то PC = PC + k + 1	1/2	—
BRCS k	Переход по переносу	Если C = 1, то PC = PC + k + 1	1/2	—
BRCC k	Переход, если нет переноса	Если C = 0, то PC = PC + k + 1	1/2	—
BREQ k	Переход по «равно»	Если Z = 1, то PC = PC + k + 1	1/2	—
BRNE k	Переход по «не равно»	Если Z = 0, то PC = PC + k + 1	1/2	—
BRSH k	Переход по «выше или равно»	Если C = 0, то PC = PC + k + 1	1/2	—
BRLO k	Переход по «меньше»	Если C = 1, то PC = PC + k + 1	1/2	—
BRMI	Переход по «отрицательное значение»	Если N = 1, то PC = PC + k + 1	1/2	—

Продолжение таблицы 3.7

Мнемоника	Описание	Операция	Циклы	Флаги
BRPL	Переход по «положительное значение»	Если N = 0, то PC = PC + k + 1	1/2	—
BRGE	Переход по «больше или равно» (числа со знаком)	Если (N ⊕ V) = 0, то PC = PC + k + 1	1/2	—
BRLT	Переход по «меньше нуля» (числа со знаком)	Если (N ⊕ V) = 1, то PC = PC + k + 1	1/2	—
BRHS	Переход по половинному переносу	Если H = 1, то PC = PC + k + 1	1/2	—
BRHC	Переход, если нет половинного переноса	Если H = 0, то PC = PC + k + 1	1/2	—
BRTS	Переход, если флаг T установлен	Если T = 1, то PC = PC + k + 1	1/2	—
BRTC	Переход, если флаг T сброшен	Если T = 0, то PC = PC + k + 1	1/2	—
BRVS	Переход по переполнению доп. кода	Если V = 1, то PC = PC + k + 1	1/2	—
BRVC	Переход, если нет переполнения доп. кода	Если V = 0, то PC = PC + k + 1	1/2	—
BRID	Переход, если прерывания запрещены	Если I = 0, то PC = PC + k + 1	1/2	—
BRIE	Переход, если прерывания разрешены	Если I = 1, то PC = PC + k + 1	1/2	—

Таблица 3.8. **Группа команд управления системой**

Мнемоника	Описание	Операция	Циклы	Флаги
NOP	Нет операции	—	1	—
SLEEP	Переход в «спящий» режим	См. соответствующие параграфы	3	—
WDR	Сброс сторожевого таймера	См. соответствующие параграфы	1	—

Глава 20. **Описание команд**

В этом параграфе в алфавитном порядке перечислены все команды, поддерживаемые микроконтроллерами семейств Tiny и Mega. Для каждой команды приводится ее детальное описание. При описании команд используются обозначения, приведенные в **Табл. 3.9.**

Таблица 3.9. **Обозначения, используемые при описании команд**

Обозначение, символ	Описание
Регистр состояния	
SREG	Регистр состояния микроконтроллера
C	Флаг переноса (0-й разряд регистра SREG)
Z	Флаг нуля (1-й разряд регистра SREG)
N	Флаг отрицательного значения (2-й разряд регистра SREG)
V	Флаг переполнения дополнительного кода (3-й разряд регистра SREG)
S	Флаг знака (4-й разряд регистра SREG); $S = N \oplus V$
H	Флаг половинного переноса (5-й разряд регистра SREG)
T	Пользовательский флаг (6-й разряд регистра SREG)
I	Флаг общего разрешения прерываний (7-й разряд регистра SREG)
Регистры и операнды	
Rd	Регистр-приемник (иногда также регистр-источник) в регистровом файле
Rr	Регистр-источник в регистровом файле
K	Константа (данные)
k	Адрес — константа
b	Номер разряда РОН или РВВ (0...7)
s	Номер разряда регистра состояния SREG (0...7)
X, Y, Z	Регистры-указатели (X = R27:R26, Y = R29:R28, Z = R31:R30)
I/O	Регистр ввода/вывода
A	Адрес в пространстве ввода/вывода

Продолжение таблицы 3.9

Обозначение, символ	Описание
q	Смещение при относительной косвенной адресации (6-разрядное значение)
.	Разделитель между названием (адресом) регистра и номером разряда
[XX]	Содержимое ячейки памяти данных по адресу XX
{XX}	Содержимое ячейки памяти программ по адресу XX
Операции	
$\overline{}$	Инверсия
•	Логическое И
$\vee$	Логическое ИЛИ
$\oplus$	Исключающее ИЛИ
Система	
PC	Счетчик команд
STACK	Текущий уровень стека
SP	Указатель стека
Флаги	
$\Leftrightarrow$	Команда воздействует на флаг
0	Флаг сбрасывается командой в «0»
1	Флаг устанавливается командой в «1»
—	Команда не влияет на состояние флага

ADC Rd, Rr

Сложение двух РОН с переносом

Операция	Rd = Rd + Rr + C							
Код операции	0001 11rd dddd rrrr						1 слово (2 байта)	
Операнды	$0 \le d \le 31$, $0 \le r \le 31$							
Описание	Складывает содержимое двух регистров Rr и Rd и прибавляет содержимое флага переноса C. Результат помещается в регистр Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	$\Leftrightarrow$	$\Leftrightarrow$	$\Leftrightarrow$	$\Leftrightarrow$	$\Leftrightarrow$	$\Leftrightarrow$
Маш. циклов	1							
Tiny	Да							

Пример	Сложение двух регистровых пар R1:R0 и R3:R2
	`add r2,r0 ;Сложить младшие байты`
	`adc r3,r1 ;Сложить старшие байты с учетом переноса`							

ADD Rd, Rr

Сложение двух РОН

Операция	Rd = Rd + Rr							
Код операции	0000 11rd dddd rrrr						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ r ≤ 31							
Описание	Складывает содержимое двух регистров Rr и Rd. Результат помещается в регистр Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	⇔	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							
Пример	`add r1,r2 ;Прибавить r2 к r1 (r1 = r1 + r2)` `add r28,r28 ;Сложить r28 с самим собой (r28 = r28 + r28)`							

ADIW Rd, K

Сложение регистровой пары с константой

Операция	Rd+1:Rd = Rd+1:Rd + K							
Код операции	1001 0110 KKdd KKKK						1 слово (2 байта)	
Операнды	d ∈ {24, 26, 28, 30}, K = 0...63							
Описание	Складывает содержимое регистровой пары Rd+1:Rd с 6-разрядным числом. Результат помещается обратно в регистровую пару							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	⇔	⇔	⇔	⇔	⇔
Маш. циклов	2							
Tiny	Нет							
Пример	`adiw r24,1 ;Прибавить 1 к r25:r24` `adiw r30,63 ;Прибавить 63 к указателю Z(r31:r30)`							

AND Rd, Rr

«Логическое И» двух РОН

Операция	Rd = Rd AND Rr							
Код операции	0010 00rd dddd rrrr						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ r ≤ 31							
Описание	Выполняет операцию «Логическое И» между содержимым регистров Rd и Rr. Результат помещается в регистр Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	⇔	**0**	⇔	⇔	—
Маш. циклов	1							
Tiny	Да							
Пример	`ldi r16,1 ;Загрузить маску 0000 0001 в r16` `and r2,r16 ;Выделить 0-й разряд в r2`							

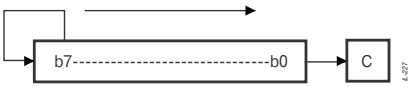
ANDI Rd, K

«Логическое И» РОН и константы

Операция	Rd = Rd AND K							
Код операции	0111 KKKK dddd KKKK						1 слово (2 байта)	
Операнды	16 ≤ d ≤ 31, 0 ≤ K ≤ 255							
Описание	Выполняет операцию «Логическое И» между содержимым регистра Rd и 8-разрядным числом. Результат помещается в регистр Rd. Команда применима только к 16 старшим РОН (R16...R31)							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	⇔	**0**	⇔	⇔	–
Маш. циклов	1							
Tiny	Да							
Пример	`andi r17,$0F  ;Обнулить старший полубайт регистра r17` `andi r18,$10  ;Выделить 4-й разряд в регистре r18`							

ASR Rd

Арифметический сдвиг вправо

Операция								
Код операции	1001 010d dddd 0101						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Сдвигает содержимое регистра Rd на 1 разряд вправо. Состояние 7-го разряда не изменяется. Значение 0-го разряда помещается в флаг C регистра SREG. Часто используется для деления чисел со знаком на два							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							

Пример	`ldi r16,$10  ;Записать число 16 в регистр r16`
	`asr r16  ;r16 = r16/2`
	`ldi r17,$FC  ;Записать число -4 в регистр r17`
	`asr r17  ;r17 = r17/2`							

BCLR s

Сброс разряда регистра SREG

Операция	SREG.s = 0							
Код операции	1001 0100 1sss 1000						1 слово (2 байта)	
Операнды	0 ≤ s ≤ 7							
Описание	Сбрасывает в «0» заданный разряд регистра SREG. Остальные разряды регистра SREG остаются без изменения							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	⇔	⇔	⇔	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							
Пример	`bclr 0 ;Сбросить флаг переноса` `bclr 7 ;Запретить прерывания`							

BLD Rd, b

Пересылка флага T в разряд РОН

Операция	Rd.b = T							
Код операции	1111 100d dddd 0bbb						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ b ≤ 7							
Описание	Копирует флаг T регистра SREG в разряд b регистра Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	1							
Tiny	Да							

Пример	Копирование разряда
	`bst r1,2 ;Сохранить 2-й разряд регистра r1 в T`
	`bld r0,4 ;Записать флаг T в 4-й разряд регистра r0`							

BRBC s, k

Переход, если разряд регистра SREG сброшен

Операция	Если SREG.s = 0, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 01kk kkkk ksss						1 слово (2 байта)	
Операнды	0 ≤ s ≤ 7, –64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет заданный разряд регистра SREG и выполняет переход, если этот разряд сброшен. Величина смещения k представляется числом в дополнительном коде							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если SREG.s = 1 2, если SREG.s = 0							
Tiny	Да							

Пример	`cpi r20,5 ;Сравнить r20 с числом 5`
	`brbc 1,noteq ;Переход, если не равно`
	`...`
	`noteq:`
	`...`							

BRBS s, k

Переход, если разряд регистра SREG установлен

Операция	Если SREG.s = 1, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 00kk kkkk ksss						1 слово (2 байта)	
Операнды	0 ≤ s ≤ 7, –64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет заданный разряд регистра SREG и выполняет переход, если этот разряд установлен. Величина смещения k представляется числом в дополнительном коде							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если SREG.s = 0 2, если SREG.s = 1							
Tiny	Да							

Пример	`bst r0,3 ;Сохранить флаг T в 3-м разряде r0`
	`brbs 1,bitset ;Переход, если флаг был установлен`
	`...`
	`bitset:`
	`...`							

BRCC k

Переход, если не было переноса

Операция	Если флаг C = 0, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 01kk kkkk k000						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг переноса (C) и выполняет переход, если этот разряд сброшен. Величина смещения k представляется числом в дополнительном коде. Эквивалентна команде BRBC 0, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если C = 1 2, если C = 0							
Tiny	Да							

Пример	`add r22,r23 ;Прибавить r23 к r22`
	`brcc nocarry ;Перейти, если не было переполнения`
	`...`
	`nocarry:`
	`...`							

BRCS k

Переход по переносу

Операция	Если флаг C = 1, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 00kk kkkk k000						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг переноса (C) и выполняет переход, если этот разряд установлен. Величина смещения k представляется числом в дополнительном коде. Эквивалентна команде BRBS 0, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если C = 0 2, если C = 1							
Tiny	Да							

Пример	`cpi r26,$56 ;Сравнить r23 с $56`
	`brcs greater ;Перейти, если r23 < $56`
	`...`
	`greater:`
	`...`							

BREQ k

Переход по «равно»

Операция	Если Rd = Rr (Z = 1), то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 00kk kkkk k001						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг нуля (Z) и выполняет переход, если этот разряд установлен. Величина смещения k представляется числом в дополнительном коде. При выполнении данной команды сразу же после команды CP, CPI, SUB или SUBI переход произойдет только в том случае, если число (со знаком или без знака), находящееся в регистре Rd, будет равно числу (со знаком или без знака), находящемуся в регистре Rr. Эквивалентна команде BRBS 1, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если Z = 0; 2, если Z = 1							
Tiny	Да							

Пример	`cp r1,r0 ;Сравнить r1 с r0`
	`breq equal ;Перейти, если r1 = r2`
	`...`
	`equal:`
	`...`							

BRGE k

Переход по «больше или равно» (для знаковых данных)

Операция	Если Rd ≥ Rr (N ⊕ V = 0), то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 01kk kkkk k100						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг знака (S) и выполняет переход, если этот разряд сброшен. Величина смещения k представляется числом в дополнительном коде. При выполнении данной команды сразу же после команды CP, CPI, SUB или SUBI, переход произойдет только в том случае, если число со знаком, находящееся в регистре Rd, будет больше или равно числу со знаком, находящемуся в регистре Rr. Эквивалентна команде BRBC 4, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если S = 1; 2, если S = 0							
Tiny	Да							

Пример	`cp r11,r12 ;Сравнить r11 с r12`
	`brge greater ;Перейти, если r1 ≥ r2`
	`...`
	`greater:`
	`...`							

BRHC k

Переход, если не было половинного переноса

Операция	Если флаг H = 0, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 01kk kkkk k101					1 слово (2 байта)		
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг половинного переноса (H) и выполняет переход, если этот разряд сброшен. Величина смещения k представляется числом в дополнительном коде. Эквивалентна команде BRBC 5, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если H = 1 2, если H = 0							
Tiny	Да							

Пример	`brne hclear ;Перейти, если флаг H сброшен`
	`...`
	`hclear:`
	`...`							

BRHS k

Переход по половинному переносу

Операция	Если флаг H = 1, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 00kk kkkk k101					1 слово (2 байта)		
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг половинного переноса (H) и выполняет переход, если этот разряд установлен. Величина смещения k представляется числом в дополнительном коде. Эквивалентна команде BRBS 5, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если H = 0 2, если H = 1							
Tiny	Да							

Пример	`brhs hset ;Перейти, если флаг H установлен`
	`...`
	`hset:`
	`...`							

BRID k

Переход, если прерывания запрещены

Операция	Если флаг I = 0, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 01kk kkkk k111						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг общего разрешения прерываний (I) и выполняет переход, если этот разряд сброшен. Величина смещения k представляется числом в дополнительном коде. Эквивалентна команде BRBC 7, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если I = 1 2, если I = 0							
Tiny	Да							

Пример	`brid intdis ;Перейти, если прерывания запрещены`
	`...`
	`intdis:`
	`...`							

BRIE k

Переход, если прерывания разрешены

Операция	Если флаг I = 1, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 00kk kkkk k111						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг общего разрешения прерываний (I) и выполняет переход, если этот разряд установлен. Величина смещения k представляется числом в дополнительном коде. Эквивалентна команде BRBS 7, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если I = 0 2, если I = 1							
Tiny	Да							

Пример	`brie inten ;Перейти, если прерывания разрешены`
	`...`
	`inten:`
	`...`							

BRLO k

Переход по «меньше» (для беззнаковых данных)

Операция	Если Rd < Rr (C = 1), то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 00kk kkkk k000						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг переноса (C) и выполняет переход, если этот разряд установлен. Величина смещения k представляется числом в дополнительном коде. При выполнении данной команды сразу же после команды CP, CPI, SUB или SUBI переход произойдет только в том случае, если беззнаковое число, находящееся в регистре Rd, будет меньше беззнакового числа, находящегося в регистре Rr. Эквивалентна команде BRBS 0, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если C = 0; 2, если C = 1							
Tiny	Да							

Пример	`eor r19,r19 ;Очистить r19`
	`loop: inc r19 ;r19 = r19 +1`
	`...`
	`cpi r19,$10 ;Сравнить r19 с $10`
	`brlo loop ;Перейти, если r19 < $10`
	`...`							

BRLT k

Переход по «меньше» (для знаковых данных)

Операция	Если Rd < Rr (N ⊕ V = 1), то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 00kk kkkk k100						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг знака (S) и выполняет переход, если этот разряд установлен. Величина смещения k представляется числом в дополнительном коде. При выполнении данной команды сразу же после команды CP, CPI, SUB или SUBI переход произойдет только в том случае, если число со знаком, находящееся в регистре Rd, будет меньше числа со знаком, находящегося в регистре Rr. Эквивалентна команде BRBS 4, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если S = 0; 2, если S = 1							
Tiny	Да							

Пример	`cp r16,r1 ;Сравнить r16 с r1`
	`brlt less ;Перейти, если r1 < r2`
	`...`
	`less:`
	`...`							

BRMI k

Переход по «отрицательное значение»

Операция	Если флаг N = 1, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 00kk kkkk k010						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг отрицательного значения (N) и выполняет переход, если этот разряд установлен. Величина смещения k представляется числом в дополнительном коде. Эквивалентна команде BRBS 2, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если N = 0 2, если N = 1							
Tiny	Да							

Пример	`subi r18,4 ;r18 = r18 - 4`
	`brmi minus ;Перейти, если результат отрицателен`
	`...`
	`minus:`
	`...`							

BRNE k

Переход по «не равно»

Операция	Если Rd ≠ Rr (Z = 0), то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 01kk kkkk k001						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг нуля (Z) и выполняет переход, если этот разряд сброшен. Величина смещения k представляется числом в дополнительном коде. При выполнении данной команды сразу же после команды CP, CPI, SUB или SUBI, переход произойдет только в том случае, если числа (со знаком или без знака), находящееся в регистрах Rd и Rr не будут равны. Эквивалентна команде BRBC 1, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если Z = 1 2, если Z = 0							
Tiny	Да							

Пример	`eor r27,r27 ;Очистить r27`
	`loop: inc r27 ;r27 = r27 + 1`
	`...`
	`cpi r27,5 ;Сравнить r27 с 5`
	`brne loop ;Перейти, если r27 ≠ 5`
	`...`							

BRPL k

Переход по «положительное значение»

Операция	Если флаг N = 0, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 01kk kkkk k010						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг отрицательного значения (N) и выполняет переход, если этот разряд сброшен. Величина смещения k представляется числом в дополнительном коде. Эквивалентна команде BRBC 2, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если N = 1 2, если N = 0							
Tiny	Да							

Пример	`subi r26,$50  ;r26 = r26 - $50`
	`brpl plus  ;Перейти, если результат положителен`
	`...`
	`plus:`
	`...`							

BRSH k

Переход по «выше или равно» (для беззнаковых данных)

Операция	Если Rd ≥ Rr (C = 0), то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 01kk kkkk k000						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг переноса (C) и выполняет переход, если этот разряд сброшен. Величина смещения k представляется числом в дополнительном коде. При выполнении данной команды сразу же после команды CP, CPI, SUB или SUBI, переход произойдет только в том случае, если беззнаковое число, находящееся в регистре Rd, будет больше или равно беззнаковому числу, находящемуся в регистре Rr. Эквивалентна команде BRBC 0, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если C = 1 2, если C = 0							
Tiny	Да							

Пример	`subi r19,4  ;r19 = r19 - 4`
	`brsh hgsm  ;Перейти, если r19 ≥ 4`
	`...`
	`hgsm:`
	`...`							

BRTC k

Переход, если флаг T сброшен

Операция	Если флаг T = 0, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 01kk kkkk k110						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг T и выполняет переход, если этот разряд сброшен. Величина смещения k представляется числом в дополнительном коде. Эквивалентна команде BRBC 6, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	1, если T = 1 2, если T = 0							
Tiny	Да							

Пример	`bst r3,5 ;Сохранить 5-й разряд r3 в флаге T`
	`brtc tclear ;Перейти, если этот разряд был сброшен`
	`...`
	`tclear:`
	`...`							

BRTS k

Переход, если флаг T установлен

Операция	Если флаг T = 1, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 00kk kkkk k110						1 слово (2 байта)	
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг T и выполняет переход, если этот разряд установлен. Величина смещения k представляется числом в дополнительном коде. Эквивалентна команде BRBS 6, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	1, если T = 0 2, если T = 1							
Tiny	Да							

Пример	`bst r3,5 ;Сохранить 5-й разряд r3 в флаге T`
	`brts tset ;Перейти, если этот разряд был установлен`
	`...`
	`tset:`
	`...`							

BRVC k

Переход, если нет переполнения дополнительного кода

Операция	Если флаг V = 0, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 01kk kkkk k011					1 слово (2 байта)		
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг переполнения дополнительного кода (V) и выполняет переход, если этот разряд сброшен. Величина смещения k представляется числом в дополнительном коде. Эквивалентна команде BRBC 3, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если V = 1 2, если V = 0							
Tiny	Да							

Пример	`add r3, r4 ;r3 = r3 + r4`
	`brvc nover ;Перейти, если этот разряд был сброшен`
	`...`
	`nover:`
	`...`							

BRVS k

Переход по переполнению дополнительного кода

Операция	Если флаг V = 1, то PC = PC + k + 1, иначе PC = PC + 1							
Код операции	1111 00kk kkkk k011					1 слово (2 байта)		
Операнды	–64 ≤ k ≤ +63							
Описание	Условный относительный переход. Проверяет флаг переполнения дополнительного кода (V) и выполняет переход, если этот разряд установлен. Величина смещения k представляется числом в дополнительном коде. Эквивалентна команде BRBS 3, k							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если V = 0 2, если V = 1							
Tiny	Да							

Пример	`add r3,r4 ;r3 = r3 + r4`
	`brvs overfl ;Перейти, если этот разряд был сброшен`
	`...`
	`overfl:`
	`...`							

BSET s

Установка разряда регистра SREG

Операция	SREG.s = 1							
Код операции	1001 0100 0sss 1000						1 слово (2 байта)	
Операнды	0 ≤ s ≤ 7							
Описание	Устанавливает заданный разряд регистра SREG							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	⇔	⇔	⇔	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							
Пример	`bset 6 ;Установить флаг T` `bset 7 ;Разрешить прерывания`							

BST Rd, b

Запись разряда РОН в флаг T

Операция	T = Rd.b							
Код операции	1111 101d dddd 0bbb						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ b ≤ 7							
Описание	Копирует разряд b регистра Rd в флаг T регистра SREG							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	⇔	–	–	–	–	–	–
Маш. циклов	1							
Tiny	Да							

Пример	Копирование бита
	`bst r1,2 ;Сохранить 2-й разряд регистра r1 в T`
	`bld r0,4 ;Записать флаг T в 4-й разряд регистра r0`							

CALL k

Абсолютный вызов подпрограммы

Операция	STACK = PC + 2; PC = k; SP = SP – 2							
Код операции	1001 010k kkkk 111k kkkk kkkk kkkk kkkk						2 слова (4 байта)	
Операнды	0 ≤ k < 64K							
Описание	Абсолютный вызов подпрограммы. Выполняет переход к подпрограмме, адрес которой задается константой k (в пределах всей памяти программ). Адрес следующей за CALL команды (2 байта) сохраняется в стеке. На практике вместо числовых значений смещения указываются метки подпрограмм (см. пример)							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	4							
Tiny	Нет							

Пример	`call routine ;Вызвать подпрограмму`
	`...`
	`routine:`
	`push r14 ;Сохранить r14`
	`...`
	`pop r14 ;Восстановить r14`
	`ret ;Возврат из подпрограммы`							

CBI A, b

Сбросить разряд PBB

Операция	I/O(A).b = 0							
Код операции	1001 1000 AAAA Abbb						1 слово (2 байта)	
Операнды	0 ≤ A ≤ 31, 0 ≤ b ≤ 7							
Описание	Сбрасывает разряд b регистра ввода/вывода, расположенного по адресу A пространства ввода/вывода. Эта команда применима только к младшим 32-м регистрам (адреса 0...31)							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Да							
Пример	`cbi $12,7 ;Сбросить 7-й разряд порта D`							

CBR Rd, K

Сброс разрядов РОН

Операция	Rd = Rd AND ($FF – K)							
Код операции	0111 $\overline{KKKK}$ dddd $\overline{KKKK}$						1 слово (2 байта)	
Операнды	16 ≤ d ≤ 31, 0 ≤ K ≤ 255							
Описание	Сбрасывает разряды в регистре Rd в соответствии с маской, задаваемой константой K. Результат помещается в регистр Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	⇔	**0**	⇔	⇔	–
Маш. циклов	1							
Tiny	Да							
Пример	`cbr r16,$F0 ;Обнулить старший полубайт регистра r16` `cbr r18,1 ;Сбросить 4-й разряд в регистре r18`							

CLC

Сброс флага переноса

Операция	C = 0							
Код операции	1001 0100 1000 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Сбрасывает в «0» флаг переноса C регистра SREG. Эквивалентна команде BCLR 0							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	**0**
Маш. циклов	1							
Tiny	Да							
Пример	`add r0,r0 ;Сложить r0 сам с собой` `clc ;Сбросить флаг переноса`							

CLH

Сброс флага половинного переноса

Операция	H = 0							
Код операции	1001 0100 1101 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Сбрасывает в «0» флаг половинного переноса H регистра SREG. Эквивалентна команде BCLR 5							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	**0**	–	–	–	–	–
Маш. циклов	1							
Tiny	Да							
Пример	`clh ;Сбросить флаг половинного переноса`							

CLI

Общее запрещение прерываний

Операция	I = 0							
Код операции	1001 0100 1111 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Сбрасывает в «0» флаг общего разрешения прерываний I регистра SREG. Эквивалентна команде BCLR 7							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	0	–	–	–	–	–	–	–
Маш. циклов	1							
Tiny	Да							

Пример	`cli ;Запретить прерывания`
	`in r11,$16 ;Прочитать состояние порта B`
	`sei ;Разрешить прерывания`							

CLN

Сброс флага отрицательного значения

Операция	N = 0							
Код операции	1001 0100 1010 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Сбрасывает в «0» флаг отрицательного значения N регистра SREG. Эквивалентна команде BCLR 2							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	**0**	–	–
Маш. циклов	1							
Tiny	Да							
Пример	`add r2,r3 ;Сложить r2 и r3` `cln ;Сбросить флаг отрицательного результата`							

CLR Rd

Очистка РОН

Операция	Rd = Rd ⊕ Rd							
Код операции	0010 01dd dddd dddd						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Сбрасывает все разряды регистра общего назначения путем выполнения операции «Исключающее ИЛИ» регистра с самим собой							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	**—**	**—**	**0**	**0**	**0**	**1**	**—**
Маш. циклов	1							
Tiny	Да							

Пример	Организация цикла с заданным числом повторений
	`clr r18 ;Очистить регистр r18`
	`loop:`
	`inc r18 ;r18 = r18 + 1`
	`...`
	`cpi r18,$50 ;Завершить цикл?`
	`brne loop`							

CLS

Сброс флага знака

Операция	S = 0							
Код операции	1001 0100 1100 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Сбрасывает в «0» флаг знака S регистра SREG. Эквивалентна команде BCLR 4							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	**—**	**—**	**0**	**—**	**—**	**—**	**—**
Маш. циклов	1							
Tiny	Да							
Пример	`add r2,r3 ;Сложить r2 и r3` `cls ;Сбросить флаг знака`							

CLT

Сброс флага T

Операция	T = 0							
Код операции	1001 0100 1110 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Сбрасывает в «0» флаг T регистра SREG. Эквивалентна команде BCLR 6							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	**0**	—	—	—	—	—	—
Маш. циклов	1							
Tiny	Да							
Пример	`clt ;Сбросить флаг T`							

CLV

Сброс флага переполнения дополнительного кода

Операция	V = 0							
Код операции	1001 0100 1011 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Сбрасывает в «0» флаг переполнения дополнительного кода V регистра SREG. Эквивалентна команде BCLR 3							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	**0**	—	—	—
Маш. циклов	1							
Tiny	Да							
Пример	`add r2,r3 ;Сложить r2 и r3` `clv ;Сбросить флаг переполнения`							

CLZ

Сброс флага нуля

Операция	Z = 0							
Код операции	1001 0100 1001 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Сбрасывает в «0» флаг нуля Z регистра SREG. Эквивалентна команде BCLR 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	**0**	—
Маш. циклов	1							
Tiny	Да							
Пример	`add r2,r3 ;Сложить r2 и r3` `clz ;Сбросить флаг нуля`							

COM Rd

Вычисление обратного кода

Операция	Rd = $FF – Rd							
Код операции	1001 010d dddd dddd						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Вычисляет обратный код числа, находящегося в регистре Rd. Результат помещается обратно в регистр Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	⇔	**0**	⇔	⇔	**1**
Маш. циклов	1							
Tiny	Да							
Пример	`com r4 ;Вычислить дополнительный код содержимого r4`							

CP Rd, Rr

Сравнение РОН

Операция	Rd – Rr							
Код операции	0001 01rd dddd rrrr						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ r ≤ 31							
Описание	Сравнивает содержимое двух регистров общего назначения путем вычитания содержимого регистра Rr из содержимого регистра Rd. Данная команда влияет только на флаги регистра состояния SREG, которые устанавливаются в соответствии с результатом вычитания. Содержимое регистров не изменяется. Как правило, данная команда используется совместно с одной из команд условного перехода							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	⇔	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							

Пример	`cp r4,r19 ;Сравнить r4 с r19 (R = r4 - r19)`
	`brne noteq ;Перейти, если r4 <> r19`
	`...`
	`noteq:`
	`...`							

CPC Rd, Rr

Сравнение РОН с учетом переноса

Операция	Rd – Rr – C							
Код операции	0000 01rd dddd rrrr						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ r ≤ 31							
Описание	Сравнивает содержимое двух регистров общего назначения путем вычитания содержимого регистра Rr и значения флага переноса (C) из содержимого регистра Rd. Данная команда влияет только на флаги регистра состояния SREG, которые устанавливаются в соответствии с результатом вычитания. Содержимое регистров не изменяется. Как правило, данная команда используется совместно с одной из команд условного перехода							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	⇔	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							

Пример	Сравнение регистровых пар r3:r2 и r1:r0
	`cp r2,r0 ;Сравнить младшие байты`
	`cpc r3,r1 ;Сравнить старшие байты`
	`brne noteq ;Перейти, если r3:r2 <> r1:r0`
	`...`
	`noteq:`
	`...`							

CPI Rd, K

Сравнение содержимого РОН с константой

Операция	Rd – K							
Код операции	0011 KKKK dddd KKKK						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ K ≤ 255							
Описание	Сравнивает содержимое регистра общего назначения Rd с константой K путем вычитания константы из содержимого регистра Rd. Данная команда влияет только на флаги регистра состояния SREG, которые устанавливаются в соответствии с результатом вычитания. Содержимое регистра Rd не изменяется. Как правило, данная команда используется совместно с одной из команд условного перехода							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	⇔	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							

Пример	`cpi r19,3 ;Сравнить r19 с числом 3 (R = r19 - 3)`
	`brne noteq ;Перейти, если r19 <> 3`
	`...`
	`noteq:`
	`...`							

CPSE Rd, Rr

Пропуск команды при равенстве двух РОН

Операция	Если Rd = Rr, то PC = PC +2 (или 3), иначе PC = PC + 1							
Код операции	0001 00rd dddd rrrr						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ r ≤ 31							
Описание	Сравнивает содержимое двух регистров общего назначения Rr и Rd и пропускает следующую команду, если в регистрах записаны одинаковые значения							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если Rd ≠ Rr (нет пропуска команды) 2, если Rd = Rr (размер пропускаемой команды — 1 слово) 3, если Rd = Rr (размер пропускаемой команды — 2 слова)							
Tiny	Да							

Пример	`inc r4 ;Увеличить r4 (r4 = r4 + 1)`
	`cpse r4,r0 ;Сравнить содержимое r4 и r0`
	`neg r4 ;Проинвертировать r4, если r4 ≠ r0`
	`...`							

DEC Rd

Декремент РОН

Операция	Rd = Rd – 1							
Код операции	1001 010d dddd 1010						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Уменьшает содержимое регистра Rd на единицу. Так как эта команда не влияет на флаг переноса C, она идеально подходит для организации счетчика числа итераций цикла при выполнении вычислений над многоразрядными числами. При работе с беззнаковыми числами для выполнения перехода в соответствии с результатом выполнения команды могут использоваться только команды условного перехода BREQ и BRNE. При работе с числами в дополнительном коде могут использоваться все команды условного перехода для знаковых проверок. Флаг V устанавливается в «1» только в том случае, если до выполнения операции в регистре находилось значение $80							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	⇔	⇔	⇔	⇔	—
Маш. циклов	1							
Tiny	Да							

Пример	`ldi r17,$10 ;Записать число 16 в регистр r17`
	`loop:`
	`add r1,r2 ;r1 = r1 + r2`
	`dec r17 ;Декрементировать r17`
	`brne loop ;Перейти, если r17 ≠ 0`
	`...`							

ELPM

Расширенная загрузка данных из памяти программ

Операция	R0 = {RAMPZ:Z}							
Код операции	1001 0101 1101 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Загружает один байт из адресного пространства памяти программ в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, содержится в регистре ввода/вывода RAMPZ и индексном регистре Z. **Эта команда поддерживается только микроконтроллерами ATmega128x!**							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	3							
Tiny	Нет							

Пример	`clr r16 ;Очистить регистр RAMPZ`
	`out RAMPZ,r16`
	`clr r31 ;Очистить старший байт индексного регистра Z`
	`ldi r30,$F0 ;Загрузить адрес в регистр Z`
	`elpm ;r0 = {$00F0}`							

ELPM Rd, Z

Расширенная загрузка данных из памяти программ

Операция	Rd = {RAMPZ:Z}							
Код операции	1001 000d dddd 0110						1 слово (2 байта)	
Операнды	$0 \le d \le 31$							
Описание	Загружает один байт из адресного пространства памяти программ в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, содержится в регистре ввода/вывода RAMPZ и индексном регистре Z. **Эта команда поддерживается только микроконтроллерами ATmega128x!**							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	3							
Tiny	Нет							

Пример	`clr r16 ;Очистить регистр RAMPZ`
	`out RAMPZ,r16`
	`clr r31 ;Очистить старший байт индексного регистра Z`
	`ldi r30,$F0 ;Загрузить адрес в регистр Z`
	`elpm r16,Z ;r16 = {$00F0}`							

ELPM Rd, Z+

Расширенная загрузка данных из памяти программ с постинкрементом

Операция	Rd = {RAMPZ:Z}, RAMPZ:Z = RAMPZ:Z + 1							
Код операции	1001 000d dddd 0111						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Загружает один байт из адресного пространства памяти программ в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, содержится в регистре ввода/вывода RAMPZ и индексном регистре Z. После пересылки байта значение указателя увеличивается на 1. **Эта команда поддерживается только микроконтроллерами ATmega128x!**							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	3							
Tiny	Нет							

Пример	`clr r16 ;Очистить регистр RAMPZ`
	`out RAMPZ,r16`
	`clr r31 ;Очистить старший байт индексного регистра Z`
	`ldi r30,$F0 ;Загрузить адрес в регистр Z`
	`elpm r16,Z+ ;r16 = {$00F0}, Z = $00F1`							

EOR Rd, Rr

«Исключающее ИЛИ» двух РОН

Операция	Rd = Rd ⊕ Rr							
Код операции	0010 01rd dddd rrrr						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ r ≤ 31							
Описание	Выполняет операцию «Исключающее ИЛИ» между регистрами Rd и Rr. Результат помещается в регистр Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	⇔	**0**	⇔	⇔	—
Маш. циклов	1							
Tiny	Да							
Пример	`eor r2,r2 ;Очистка регистра r2` `eor r0,r22 ;Побитовое «Исключающее ИЛИ» между r0 и r22`							

FMUL Rd, Rr

Умножение дробных беззнаковых чисел

Операция	R1:R0 = (Rd × Rr) << 1							
Код операции	0000 0011 0ddd 1rrr						1 слово (2 байта)	
Операнды	16 ≤ d ≤ 23, 16 ≤ r ≤ 23							
Описание	Осуществляет умножение беззнаковых дробных чисел, находящихся в регистрах Rd и Rr. Формат чисел — 1.7 (старший разряд — целая часть, 7 младших — дробная). Результат умножения (формат результата — 2.14) сдвигается влево на один разряд для приведения к формату 1.15 и заносится в регистровую пару R1:R0							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	⇔	⇔
Маш. циклов	2							
Tiny	Нет							
Пример	`fmul r23,r22  ;Умножить r23 и r22` `movw r22,r0  ;Скопировать результат обратно в r23:r22`							

FMULS Rd, Rr

Умножение дробных чисел со знаком

Операция	R1:R0 = (Rd × Rr) << 1							
Код операции	0000 0011 1ddd 0rrr						1 слово (2 байта)	
Операнды	16 ≤ d ≤ 23, 16 ≤ r ≤ 23							
Описание	Осуществляет умножение дробных чисел со знаком, находящихся в регистрах Rd и Rr. Формат чисел — 1.7 (старший разряд — целая часть, 7 младших разрядов — дробная). Результат умножения (формат результата — 2.14) сдвигается влево на один разряд для приведения к формату 1.15 и заносится в регистровую пару R1:R0							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	⇔	⇔
Маш. циклов	2							
Tiny	Нет							
Пример	`fmuls r23,r22  ;Умножить r23 и r22` `movw r22,r0  ;Скопировать результат обратно в r23:r22`							

FMULSU Rd, Rr

Умножение дробного беззнакового числа и дробного числа со знаком

Операция	R1:R0 = (Rd × Rr) << 1							
Код операции	0000 0011 1ddd 1rrr						1 слово (2 байта)	
Операнды	16 ≤ d ≤ 23, 16 ≤ r ≤ 23							
Описание	Осуществляет умножение дробных чисел, находящихся в регистрах Rd (число со знаком) и Rr (число без знака). Формат чисел — 1.7 (старший разряд — целая часть, 7 младших разрядов — дробная). Результат умножения (формат результата соответствует 2.14) сдвигается влево на один разряд для приведения к формату 1.15 и заносится в регистровую пару R1:R0							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	⇔	⇔
Маш. циклов	2							
Tiny	Нет							
Пример	`fmulsu r23,r22 ;Умножить r23 и r22` `movw r22,r0 ;Скопировать результат обратно в r23:R22`							

ICALL

Косвенный вызов подпрограммы

Операция	STACK = PC + 1; PC = Z; SP = SP – 2							
Код операции	1001 0101 0000 1001						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Косвенный вызов подпрограммы. Выполняет переход к подпрограмме, адрес которой находится в регистре Z. Адрес следующей за ICALL команды (2 байта) сохраняется в стеке							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	3							
Tiny	Нет							

Пример	`mov r30,r0 ;Задать смещение`
	`icall ;Вызвать подпрограмму, адрес которой`
	`;находится в регистрах r31:r30`							

IJMP

Косвенный переход

Операция	PC = Z							
Код операции	1001 0100 0000 1001						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Косвенный безусловный переход. Выполняет переход по адресу, находящемуся в регистре Z							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	2							
Tiny	Нет							
Пример	`mov r30,r0 ;Задать смещение` `ijmp ;Перейти по адресу r31:r30`							

IN Rd, A

Пересылка значения из РВВ в РОН

Операция	Rd = I/O(A)							
Код операции	`1011 0AAd dddd AAAA`						1 слово (2 байта)	
Операнды	$0 \le d \le 31$, $0 \le A \le 63$							
Описание	Пересылает содержимое регистра ввода/вывода A в регистр общего назначения Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1							
Tiny	Да							

Пример	`in r25,$16 ;Прочитать содержимое порта B`
	`cpi r25,4 ;Сравнить содержимое с константой`
	`breq exit ;Перейти, если r25=4`
	`...`
	`exit:`
	`...`							

INC Rd

Инкремент РОН

Операция	Rd = Rd + 1							
Код операции	1001 010d dddd 0011						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Увеличивает содержимое регистра Rd на единицу. Так как эта команда не влияет на флаг переноса C, она идеально подходит для организации счетчика числа итераций цикла при выполнении вычислений над многоразрядными числами. При работе с беззнаковыми числами для выполнения перехода в соответствии с результатом выполнения команды могут использоваться только команды условного перехода BREQ и BRNE. При работе с числами в дополнительном коде могут использоваться все команды условного перехода для знаковых проверок. Флаг V устанавливается в «1» только в том случае, если до выполнения операции в регистре находилось значение $7F							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	⇔	⇔	⇔	⇔	–
Маш. циклов	1							
Tiny	Да							

Пример

```
clr r22         ;Очистить регистр r22
loop:
inc r22         ;r22=r22+1
...
cpi r22,$4F     ;
brne loop       ;Продолжать цикл, если r22≠$4F
...
```

JMP k

Абсолютный безусловный переход

Операция	PC = k							
Код операции	1001 010k kkkk 110k kkkk kkkk kkkk kkkk						2 слова (4 байта)	
Операнды	-2047 ≤ k ≤ 4M							
Описание	Команда относительного безусловного перехода. Выполняет переход по адресу, равному сумме содержимого счетчика команд и константы k. На практике вместо числовых значений смещения используются метки (см. пример)							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	3							
Tiny	Нет							

Пример	`cpi r16,$42 ;Сравнить r16 с числом $42`
	`brne error ;Перейти, если r16 ≠ $42`
	`rjmp ok ;Безусловный переход`
	`error:`
	`add r16, r17 ;Прибавить r16 к r17`
	`inc r16 ;r16 = r16 + 1`
	`ok:`
	`...`							

LD Rd, X

Косвенное чтение памяти данных

Операция	Rd = [X]							
Код операции	1001 000d dddd 1100						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Загружает один байт из адресного пространства памяти данных в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре X							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r27 ;Очистить старший байт индексного регистра`
	`ldi r26,$60 ;Загрузить младший байт адреса`
	`ld r1,X ;r1 = [$0060]`							

LD Rd, X+

Косвенное чтение памяти данных с постинкрементом

Операция	Rd = [X], X = X + 1							
Код операции	1001 000d dddd 1101						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, для d = 26 или 27 результат операции неопределен							
Описание	Загружает один байт из адресного пространства памяти данных в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре X. После пересылки байта содержимое регистра X увеличивается на 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r27 ;Очистить старший байт индексного регистра`
	`ldi r26,$60 ;Загрузить младший байт адреса`
	`ld r1,X+ ;r1 = [$0060]`
	`in r1,r26 ;В регистре r1 - $61`							

LD Rd, –X

Косвенное чтение памяти данных с преддекрементом

Операция	X = X – 1, Rd = [X]							
Код операции	1001 000d dddd 1110						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, для d = 26 или 27 результат операции неопределен							
Описание	Загружает один байт из адресного пространства памяти данных в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре X, причем перед обращением к памяти данных содержимое регистра X уменьшается на 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r27 ;Очистить ст. байт индексного регистра`
	`ldi r26,$63 ;Загрузить мл. байт адреса`
	`ld r3,-X ;r3 = [$0062]`							

LD Rd, Y

Косвенное чтение памяти данных

Операция	Rd = [Y]							
Код операции	1000 000d dddd 1000						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Загружает один байт из адресного пространства памяти данных в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Y							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r29 ;Очистить ст. байт индексного регистра`
	`ldi r28,$60 ;Загрузить мл. байт адреса`
	`ld r1,Y ;r1 = [$0060]`							

LD Rd, Y+

Косвенное чтение памяти данных с постинкрементом

Операция	Rd =[Y], Y = Y + 1							
Код операции	1001 000d dddd 1001						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, для d = 28 или 29 результат операции не определен							
Описание	Загружает один байт из адресного пространства памяти данных в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Y. После пересылки байта содержимое регистра Y увеличивается на 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r29 ;Очистить ст. байт индексного регистра`
	`ldi r28,$60 ;Загрузить мл. байт адреса`
	`ld r1,Y+ ;r1 = [$0060]`
	`in r2,r28 ;В регистре r2 - $61`							

LD Rd, –Y

Косвенное чтение памяти данных с преддекрементом

Операция	Y = Y – 1, Rd = [Y]							
Код операции	1001 000d dddd 1010						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, для d = 28 или 29 результат операции неопределен							
Описание	Загружает один байт из адресного пространства памяти данных в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Y, причем перед обращением к памяти данных содержимое регистра Y уменьшается на 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r29 ;Очистить старший байт индексного регистра`
	`ldi r28,$63 ;Загрузить младший байт адреса`
	`ld r3,-Y ;r3 = [$0062]`							

LD Rd, Z

Косвенное чтение памяти данных

Операция	Rd = [Z]							
Код операции	1000 000d dddd 0000						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Загружает один байт из адресного пространства памяти данных в регистр общего назначения Rd. Для моделей семейства Tiny, рассматриваемых в книге, в адресное пространство памяти данных входит только регистровый файл, для остальных — регистровый файл, память ввода/вывода и ОЗУ. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Z							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Да							

Пример	`clr r31 ;Очистить старший байт индексного регистра`
	`ldi r30,$60 ;Загрузить младший байт адреса`
	`ld r1,Z ;r1 = $0060]`							

LD Rd, Z+

Косвенное чтение памяти данных с постинкрементом

Операция	Rd = [Z], Z = Z + 1							
Код операции	1001 000d dddd 0001						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, для d = 30 или 31 результат операции неопределен							
Описание	Загружает один байт из адресного пространства памяти данных в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Z. После пересылки байта содержимое регистра Z увеличивается на 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r31 ;Очистить старший байт индексного регистра`
	`ldi r30,$60 ;Загрузить младший байт адреса`
	`ld r1,Z+ ;r1 = [$0060]`
	`in r2,r30 ;В регистре r2 - $61`							

LD Rd, –Z

Косвенное чтение памяти данных с преддекрементом

Операция	Z = Z – 1, Rd = [Z]							
Код операции	1001 000d dddd 0010						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, для d = 30 или 31 результат операции не определен							
Описание	Загружает один байт из адресного пространства памяти данных в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Z, причем перед обращением к памяти данных содержимое регистра Z уменьшается на 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r31 ;Очистить старший байт индексного регистра`
	`ldi r30,$63 ;Загрузить младший байт адреса`
	`ld r3,-Z ;r3 = [$0062]`							

LDD Rd, Y+q

Косвенное относительное чтение памяти данных

Операция	Rd = [Y + q]							
Код операции	10q0 qq0d dddd 1qqq						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ q ≤ 63							
Описание	Загружает один байт из адресного пространства памяти данных в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, получается суммированием содержимого индексного регистра Y и константы q. Содержимое индексного регистра не изменяется							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r29 ;Очистить старший байт индексного регистра`
	`ldi r28,$63 ;Загрузить младший байт адреса`
	`ldd r4,Y+2 ;r4 = [$0065]`							

LDD Rd, Z+q

Косвенное относительное чтение памяти данных

Операция	Rd = [Z + q]							
Код операции	10q0 qq0d dddd 0qqq						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ q ≤ 63							
Описание	Загружает один байт из адресного пространства памяти данных в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, получается суммированием содержимого индексного регистра Z и константы q. Содержимое индексного регистра не изменяется							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r31 ;Очистить старший байт индексного регистра`
	`ldi r30,$63 ;Загрузить младший байт адреса`
	`ldd r4,Z+2 ;r4 = [$0065]`							

LDI Rd, K

Загрузка константы в РОН

Операция	Rd = K							
Код операции	1110 KKKK dddd KKKK						1 слово (2 байта)	
Операнды	16 ≤ d ≤ 31, 0 ≤ k ≤ 255							
Описание	Загружает 8-разрядное число в регистр общего назначения Rd. Данная команда применима только к старшей половине РОН (адреса 16...31)							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1							
Tiny	Да							

Пример	`clr r31 ;Очистить ст. байт индексного регистра Z`
	`ldi r30,$F0 ;Загрузить адрес в регистр Z`
	`lpm ;Загрузить константу из памяти программ`
	`;по адресу $00F0`							

LDS Rd, k

Непосредственная загрузка из памяти данных

Операция	Rd = [k]							
Код операции	1001 000d dddd 0000 kkkk kkkk kkkk kkkk						2 слова (4 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ k ≤ 65535							
Описание	Загружает один байт из адресного пространства памяти данных в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, задается константой K							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	2							
Tiny	Нет							

Пример	`lds r2,$FF00 ;r2 = [$FF00]`
	`add r2,r1 ;r2 = r2 + r1`
	`sts $FF00,r2 ;Записать результат по тому же адресу`							

LPM

Загрузка данных из памяти программ

Операция	R0 = {Z}							
Код операции	1001 0101 1100 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Загружает один байт из адресного пространства памяти программ в регистр общего назначения R0. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Z							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	3							
Tiny	Да							

Пример	`clr r31 ;Очистить ст. байт индексного регистра Z`
	`ldi r30,$F0 ;Загрузить адрес в регистр Z`
	`lpm ;r0 = {$00F0}`							

LPM Rd, Z

Загрузка данных из памяти программ

Операция	Rd = {Z}							
Код операции	1001 000d dddd 0100						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Загружает один байт из адресного пространства памяти программ в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Z							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	3							
Tiny	Нет							

Пример	`clr r31 ;Очистить ст. байт индексного регистра Z`
	`ldi r30,$F0 ;Загрузить адрес в регистр Z`
	`lpm r16,Z ;r16 = {$00F0}`							

LPM Rd, Z+

Загрузка данных из памяти программ с постинкрементом

Операция	Rd = {Z}, Z=Z+1							
Код операции	1001 000d dddd 0101						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Загружает один байт из адресного пространства памяти программ в регистр общего назначения Rd. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Z. После пересылки байта содержимое регистра Z увеличивается на 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	3							
Tiny	Нет							

Пример	`clr r31 ;Очистить ст. байт индексного регистра Z`
	`ldi r30,$F0 ;Загрузить адрес в регистр Z`
	`lpm r16,Z+ ;r16 = {$00F0}, Z = $00F1`							

LSL Rd

Логический сдвиг влево

Операция								
Код операции	1000 11dd dddd dddd						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Сдвигает все разряды регистра Rd влево. Разряд b0 сбрасывается в «0», а разряд b7 загружается в флаг C регистра SREG. Эквивалентна команде ADD Rd, Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	⇔	⇔	**0**	⇔	⇔
Маш. циклов	1							
Tiny	Да							
Пример	`add r0,r4 ;r0 = r0 + r4` `lsl r0 ;r0 = r0 * 2`							

LSR Rd

Логический сдвиг вправо

Операция								
Код операции	1001 010d dddd 0110						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Сдвигает все разряды регистра Rd вправо. Разряд b7 сбрасывается в «0», а разряд b0 загружается в флаг C регистра SREG							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	⇔	⇔	**0**	⇔	⇔
Маш. циклов	1							
Tiny	Да							
Пример	`add r0,r4 ;r0 = r0 + r4` `lsr r0 ;r0 = r0/2`							

MOV Rd, Rr

Пересылка между РОН

Операция	Rd = Rr							
Код операции	0010 11rd dddd rrrr						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ r ≤ 31							
Описание	Копирует содержимое регистра Rr в регистр Rd. Регистр-источник (Rr) не изменяется							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1							
Tiny	Да							

Пример	`mov r16,r0 ;Переслать содержимое r0 в r16`
	`call check ;Вызвать подпрограмму`
	`...`
	`check:`
	`cpi r16, $/11 ;Сравнить r16 с $11`
	`...`
	`ret ;Вернуться из подпрограммы`							

MOVW Rd, Rr

Пересылка между парами РОН

Операция	Rd + 1:Rd = Rr + 1:Rr							
Код операции	0000 0001 dddd rrrr						1 слово (2 байта)	
Операнды	d ∈ {0, 2,..., 30}, r ∈ {0, 2,..., 30}							
Описание	Копирует содержимое регистровой пары Rr + 1:Rr в регистровую пару Rd + 1:Rd. Регистры-источники (Rr + 1 и Rr) не изменяются							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1							
Tiny	Нет							
Пример	`movw r16,r0 ;Переслать r1:r0 в r17:r16`							

MUL Rd, Rr

Умножение беззнаковых чисел

Операция	R1:R0 = Rd × Rr							
Код операции	1001 11rd dddd rrrr						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ r ≤ 31							
Описание	Осуществляет умножение беззнаковых чисел, находящихся в регистрах Rd и Rr. Результат умножения заносится в регистровую пару R1:R0							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	⇔	⇔
Маш. циклов	2							
Tiny	Нет							
Пример	`mul r5,r4 ;Умножить r5 и r4` `movw r5,r0 ;Скопировать результат обратно в r5:r4`							

MULS Rd, Rr

Умножение чисел со знаком

Операция	R1:R0 = Rd × Rr							
Код операции	0000 0010 dddd rrrr						1 слово (2 байта)	
Операнды	16 ≤ d ≤ 31, 16 ≤ r ≤ 31							
Описание	Осуществляет умножение чисел со знаком, находящихся в регистрах Rd и Rr. Результат умножения заносится в регистровую пару R1:R0							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	⇔	⇔
Маш. циклов	2							
Tiny	Нет							
Пример	`muls r16,r17  ;Умножить r16 и r17` `movw r16,r0  ;Скопировать результат обратно в r17:r16`							

MULSU Rd, Rr

Умножение беззнакового числа и числа со знаком

Операция	R1:R0 = Rd × Rr							
Код операции	0000 0011 0ddd 0rrr						1 слово (2 байта)	
Операнды	16 ≤ d ≤ 23, 16 ≤ r ≤ 23							
Описание	Осуществляет умножение чисел, находящихся в регистрах Rd (число со знаком) и Rr (число без знака). Результат умножения заносится в регистровую пару R1:R0							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	⇔	⇔
Маш. циклов	2							
Tiny	Нет							
Пример	`mulsu r16,r17  ;Умножить r16 и r17` `movw r16,r0  ;Скопировать результат обратно в r17:r16`							

NEG Rd

Вычисление дополнительного кода

Операция	Rd = $00 – Rd							
Код операции	1001 010d dddd 0001						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Вычисляет дополнительный код числа, находящегося в регистре Rd. Результат помещается обратно в регистр Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	⇔	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							

Пример	`sub r11,r0 ;Вычесть r0 из r11 (r11 = r11 - r0)`
	`brpl positive ;Перейти, если результат положителен`
	`neg r11 ;Вычислить дополнительный код числа`
	`positive:`
	`...`							

NOP

Пустая команда

Операция	Нет операции							
Код операции	0000 0000 0000 0000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Выполняет пустой машинный цикл							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	1							
Tiny	Да							

Пример	`clr r16 ;Очистить регистр r16`
	`ser r17 ;Установить регистр r17`
	`out $18,r16 ;Записать нули в порт B`
	`nop ;Ждать один машинный цикл`
	`out $18,r17 ;Записать единицы в порт B`							

OR Rd, Rr

«Логическое ИЛИ» двух РОН

Операция	Rd = Rd ∨ Rr							
Код операции	0010 10rd dddd rrrr						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ r ≤ 31							
Описание	Выполняет операцию «Логическое ИЛИ» между регистрами Rd и Rr. Результат помещается в регистр Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	⇔	**0**	⇔	⇔	–
Маш. циклов	1							
Tiny	Да							

```
or r15,r16     ;Поразрядное «ИЛИ» r15 и r16
bst r15,6      ;Записать 6-й разряд регистра r15 в флаг T
brts ok        ;Перейти, если флаг T равен «1»
...
ok:
...
```

ORI Rd, K

«Логическое ИЛИ» РОН и константы

Операция	Rd = Rd ∨ K							
Код операции	0110 KKKK dddd KKKK						1 слово (2 байта)	
Операнды	16 ≤ d ≤ 31, 0 ≤ K ≤ 255							
Описание	Выполняет операцию «Логическое ИЛИ» между регистром Rd и константой K. Результат помещается в регистр Rd. Команда применима только к 16 старшим РОН (R16...R31)							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	⇔	**0**	⇔	⇔	–
Маш. циклов	1							
Tiny	Да							

```
ori r17,$0F    ;Установить старший полубайт регистра r17
ori r18,1      ;Установить 0-й разряд регистра r18
```

OUT A, Rr

Пересылка значения из РОН в РВВ

Операция	I/O(A) = Rr							
Код операции	1011 1AAr rrrr AAAA						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31, 0 ≤ A ≤ 63							
Описание	Пересылает содержимое регистра общего назначения Rr в регистр ввода/вывода A							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1							
Tiny	Да							

Пример	`clr r16 ;Очистить регистр r16`
	`ser r17 ;Установить регистр r17`
	`out $18,r16 ;Записать нули в порт B`
	`nop ;Ждать один машинный цикл`
	`out $18,r17 ;Записать единицы в порт B`							

POP Rd

Извлечение байта из стека

Операция	SP = SP + 1, Rd = STACK							
Код операции	1001 000d dddd 1111						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Загружает 1 байт из стека в регистр общего назначения Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	2							
Tiny	Нет							

Пример	`call routine ;Вызов подпрограммы`
	`...`
	`routine:`
	`push r14 ;Сохранить r14 в стеке`
	`push r15 ;Сохранить r15 в стеке`
	`...`
	`pop r15 ;Восстановить r15 из стека`
	`pop r14 ;Восстановить r14 из стека`
	`ret ;Возврат из подпрограммы`							

PUSH Rr

Сохранение байта в стеке

Операция	STACK = Rr , SP = SP – 1							
Код операции	1001 001r rrrr 1111						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31							
Описание	Сохраняет содержимое регистра общего назначения Rr в стеке							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`call routine ;Вызов подпрограммы`
	`...`
	`routine:`
	`push r14 ;Сохранить r14 в стеке`
	`push r15 ;Сохранить r15 в стеке`
	`...`
	`pop r15 ;Восстановить r15 из стека`
	`pop r14 ;Восстановить r14 из стека`
	`ret ;Возврат из подпрограммы`							

RCALL k

Относительный вызов подпрограммы

Операция	STACK = PC + 1; PC = PC + k + 1; SP = SP – 2							
Код операции	1101 kkkk kkkk kkkk						1 слово (2 байта)	
Операнды	– 2047 ≤ k ≤ 2047							
Описание	Относительный вызов подпрограммы. Выполняет переход к подпрограмме, адрес которой получается сложением содержимого счетчика команд с константой k. Адрес следующей за RCALL команды (2 байта) сохраняется в стеке. На практике вместо числовых значений смещения указываются метки подпрограмм (см. пример)							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	3							
Tiny	Да							

Пример	`rcall routine ;Вызвать подпрограмму`
	`...`
	`routine:`
	`push r14 ;Сохранить r14`
	`...`
	`pop r14 ;Восстановить r14`
	`ret ;Возврат из подпрограммы`							

RET

Возврат из подпрограммы

Операция	SP = SP + 2; PC = STACK							
Код операции	1001 0101 0000 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Возврат из подпрограммы. Выполняет возврат в то место, откуда подпрограмма была вызвана							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	4							
Tiny	Да							

Пример	`rcall routine ;Вызвать подпрограмму`
	`...`
	`routine:`
	`push r14 ;Сохранить r14`
	`...`
	`pop r14 ;Восстановить r14`
	`ret ;Возврат из подпрограммы`							

RETI

Возврат из подпрограммы обработки прерывания

Операция	SP = SP + 2; PC = STACK							
Код операции	1001 0101 0001 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Возврат из подпрограммы обработки прерывания. Выполняет возврат в то место, в котором выполнение программы было прервано в результате возникновения прерывания. Следует обратить внимание, что контекст программы (регистр состояния SREG) не сохраняется при вызове подпрограммы обработки прерывания и соответственно не восстанавливается при выходе из нее. В связи с этим сохранение и восстановление этого регистра необходимо выполнять самостоятельно							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	1	—	—	—	—	—	—	—
Маш. циклов	4							
Tiny	Да							

Пример	`...`
	`extint:`
	`push r0 ;Сохранить r0`
	`...`
	`pop r0 ;Восстановить r0`
	`reti ;Возврат из подпрограммы`							

RJMP k

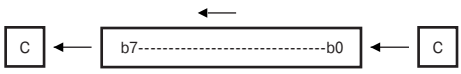
Относительный безусловный переход

Операция	PC = PC + k + 1;							
Код операции	1100 kkkk kkkk kkkk						1 слово (2 байта)	
Операнды	-2047 ≤ k ≤ 2047							
Описание	Команда относительного безусловного перехода. Выполняет переход по адресу, равному сумме содержимого счетчика команд и константы k. На практике вместо числовых значений смещения используются метки (см. пример)							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	2							
Tiny	Да							

Пример	`cpi r16,$42 ;Сравнить r16 с числом $42`
	`brne error ;Перейти, если r16 ≠ $42`
	`rjmp ok ;Безусловный переход`
	`error:`
	`add r16, r17 ;Прибавить r16 к r17`
	`inc r16 ;r16 = r16 + 1`
	`ok:`
	`...`							

ROL Rd

Сдвиг влево через перенос

Операция								
Код операции	0001 11dd dddd dddd						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Сдвигает содержимое регистра Rd влево на один разряд. В разряд b0 заносится содержимое флага C регистра SREG, а разряд b7 загружается в флаг C. В комбинации с командой LSL данная команда может использоваться для умножения многобайтных чисел (как знаковых, так и беззнаковых) на два. Эквивалентна команде ADC Rd,Rd. Значение флага V равно «Исключающему ИЛИ» флагов N и C после сдвига							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	⇔	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							

Пример	Умножить регистровую пару r19:r18 на 2
	`lsl r18 ;r18 = r18 * 2`
	`rol r19 ;r19:r18 — 16-разрядное целое`							

ROR Rd

Сдвиг вправо через перенос

Операция								
Код операции	1001 010d dddd 0111						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Сдвигает содержимое регистра Rd вправо на один разряд. В разряд b7 заносится содержимое флага C регистра SREG, а разряд b0 загружается в флаг C. В комбинации с командой ASR данная команда может использоваться для деления многобайтных знаковых чисел на два. А в комбинации с командой LSR — для деления многобайтных беззнаковых чисел на два. Значение флага V равно «Исключающему ИЛИ» флагов N и C после сдвига							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							

Пример	`lsr r19 ;Разделить регистровую пару r19:r18 на 2`
	`ror r18 ;r19:r18 – 16-разрядное целое без знака`
	`asr r17 ;Разделить регистровую пару r17:r16 на 2`
	`ror r16 ;r17:r16 – 16-разрядное целое со знаком`							

SBC Rd, Rr

Вычитание с заемом

Операция	Rd = Rd – Rr – C							
Код операции	0000 10rd dddd rrrr						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ r ≤ 31							
Описание	Вычитает из регистра Rd содержимое регистра Rr. Если флаг переноса C установлен, полученная разность уменьшается на 1. Результат помещается в регистр Rd. Если результат вычитания не равен нулю, флаг нуля Z сбрасывается в «0», в противном случае он остается без изменений							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	⇔	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							

Пример	Вычитание регистровой пары R1:R0 из R3:R2
	`sub r2,r0 ;Вычесть младшие байты`
	`sbc r3,r1 ;Вычесть старшие байты с учетом переноса`							

SBCI Rd, K

Вычитание константы из РОН с заемом

Операция	Rd = Rd – K – C							
Код операции	0100 KKKK dddd KKKK						1 слово (2 байта)	
Операнды	16 ≤ d ≤ 31, 0 ≤ K ≤ 255							
Описание	Вычитает из регистра Rd значение константы K. Если флаг переноса C установлен, полученная разность уменьшается на 1. Результат помещается в регистр Rd. Если результат вычитания не равен нулю, флаг нуля Z сбрасывается в «0», в противном случае он остается без изменений							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	⇔	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							

Пример	Вычитание двухбайтного числа $4F23 из регистровой пары R17:R16
	`subi r16,$23 ;Вычесть младший байт`
	`sbci r17,$4F ;Вычесть старший байт с учетом переноса`							

SBI A, b

Установить разряд РВВ

Операция	I/O(A).b = 1							
Код операции	1001 1010 AAAA Abbb						1 слово (2 байта)	
Операнды	0 ≤ A ≤ 31, 0 ≤ b ≤ 7							
Описание	Устанавливает разряд b регистра ввода/вывода, расположенного по адресу A пространства ввода/вывода. Эта команда применима только к младшим 32-м регистрам (адреса 0...31)							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Да							

Пример	`out $1E,r0 ;Задать адрес ячейки EEPROM`
	`sbi $1C,0 ;Установить запрос на чтение`
	`in r1,$1D ;Считать данные из EEPROM`							

SBIC A, b

Пропустить команду, если разряд PBB сброшен

Операция	Если I/O(A).b = 0, то PC = PC + 2 (3), иначе PC = PC + 1							
Код операции	1001 1001 AAAA Abbb						1 слово (2 байта)	
Операнды	0 ≤ A ≤ 31, 0 ≤ b ≤ 7							
Описание	Проверяет состояние разряда b регистра ввода/вывода A. Если разряд сброшен, команда, следующая за «SBIC A,b», пропускается. Эта команда применима только к младшим 32-м регистрам (адреса 0...31)							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если I/O(A).b = 1 (нет пропуска команды) 2, если I/O(A).b = 0 (размер пропускаемой команды — 1 слово) 3, если I/O(A).b = 0 (размер пропускаемой команды — 2 слова)							
Tiny	Да							

Пример	`e2wait:`
	`sbic $1C,1 ;Пропустить команду, если флаг EEWE сброшен`
	`rjmp e2wait ;Запись в EEPROM еще не закончилась`
	`...`							

SBIS A, b

Пропустить команду, если разряд PBB установлен

Операция	Если I/O(A).b = 1 то PC = PC + 2 (3), иначе PC = PC + 1							
Код операции	1001 1011 AAAA Abbb						1 слово (2 байта)	
Операнды	0 ≤ A ≤ 31, 0 ≤ b ≤ 7							
Описание	Проверяет состояние разряда b регистра ввода/вывода A. Если разряд установлен, команда, следующая за «SBIS A,b», пропускается. Эта команда применима только к младшим 32-м регистрам (адреса 0...31)							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если I/O(A).b = 0 (нет пропуска команды) 2, если I/O(A).b = 1 (размер пропускаемой команды — 1 слово) 3, если I/O(A).b = 1 (размер пропускаемой команды — 2 слова)							
Tiny	Да							

Пример	`waitset:`
	`sbis $1D,0 ;Пропустить следующую команду,`
	`;если 0-й разряд`
	`;порта D установлен`
	`rjmp waitset ;Ждать, пока разряд не будет установлен`							

SBIW Rd, K

Вычитание константы из регистровой пары

Операция	Rd + 1:Rd = Rd + 1:Rd – K							
Код операции	1001 0111 KKdd KKKK						1 слово (2 байта)	
Операнды	d ∈ {24, 26, 28, 30}, 0 ≤ K ≤ 63							
Описание	Вычитает из регистровой пары Rd + 1:Rd значение константы K. Результат помещается обратно в регистровую пару. Команда применима только к 4-м старшим регистровым парам из регистров общего назначения							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	⇔	⇔	⇔	⇔	⇔
Маш. циклов	2							
Tiny	Нет							

Пример	Вычитание двухбайтного числа $4F23 из регистровой пары R17:R16
	`sbiw r24,1 ;Вычесть единицу из r25:r24`
	`sbiw r28,63 ;Вычесть 63 из индексного регистра Y`							

SBR Rd, K

Установка разрядов РОН

Операция	Rd = Rd ∨ K							
Код операции	0110 KKKK dddd KKKK						1 слово (2 байта)	
Операнды	16 ≤ d ≤ 31, 0 ≤ K ≤ 255							
Описание	Устанавливает отдельные разряды регистра Rd путем выполнения операции «Логическое ИЛИ» между содержимым регистра Rd и маской, задаваемой константой K. Результат помещается в регистр Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	⇔	**0**	⇔	⇔	–
Маш. циклов	1							
Tiny	Да							
Пример	`sbr r16,3 ;Установить разряды 0 и 1 регистра r16` `sbr r18,$F0 ;Установить 4 старших разряда регистра r18`							

SBRC Rr, b

Пропустить команду, если разряд РОН сброшен

Операция	Если Rr.b = 0 то PC = PC +2 (3), иначе PC = PC + 1							
Код операции	1111 110r rrrr 0bbb						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31, 0 ≤ b ≤ 7							
Описание	Проверяет состояние разряда b регистра общего назначения Rr. Если разряд сброшен, команда, следующая за «SBRC Rr,b», пропускается							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если Rr.b = 1 (нет пропуска команды) 2, если Rr.b = 0 (размер пропускаемой команды – 1 слово) 3, если Rr.b = 0 (размер пропускаемой команды – 2 слова)							
Tiny	Да							

Пример	`sub r0,r1 ;Вычесть r1 из r0`
	`sbrc r0,7 ;Пропустить команду, если r0,7 = «0»`
	`sub r0,r1 ;Выполняется только, если r0,7 = «1»`							

SBRS Rr ,b

Пропустить команду, если разряд РОН установлен

Операция	Если Rr.b = 1 то PC = PC + 2 (3), иначе PC = PC + 1							
Код операции	1111 111r rrrr 0bbb						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31, 0 ≤ b ≤ 7							
Описание	Проверяет состояние разряда b регистра общего назначения Rr. Если разряд установлен, команда, следующая за «SBRS Rr,b», пропускается							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1, если Rr.b = 0 (нет пропуска команды) 2, если Rr.b = 1 (размер пропускаемой команды — 1 слово) 3, если Rr.b = 1 (размер пропускаемой команды — 2 слова)							
Tiny	Да							

Пример	`sub r0,r1 ;Вычесть r1 из r0`
	`sbrs r0,7 ;Пропустить след. команду, если r0,7 = «0»`
	`neg r0 ;Выполняется, только если r0,7 = «1»`							

SEC

Установка флага переноса

Операция	C = 1							
Код операции	1001 0100 0000 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Устанавливает в «1» флаг переноса C регистра SREG. Эквивалентна команде BSET 0							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	**1**
Маш. циклов	1							
Tiny	Да							
Пример	`sec ;Установить флаг переноса` `addc r0,r1 ;r0 = r0 + r1 + 1`							

SEH

Установка флага половинного переноса

Операция	H = 1							
Код операции	1001 0100 0101 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Устанавливает в «1» флаг половинного переноса H регистра SREG. Эквивалентна команде BSET 5							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	**1**	—	—	—	—	—
Маш. циклов	1							
Tiny	Да							
Пример	`slh ;Установить флаг половинного переноса`							

SEI

Общее разрешение прерываний

Операция	I = 1							
Код операции	1001 0100 0111 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Устанавливает в «1» флаг общего разрешения прерываний I регистра SREG. Эквивалентна команде BSET 7							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	1	—	—	—	—	—	—	—
Маш. циклов	1							
Tiny	Да							

Пример	`cli ;Запретить прерывания`
	`in r13,$16 ;Прочитать состояние порта B`
	`sei ;Разрешить прерывания`							

SEN

Установка флага отрицательного значения

Операция	N = 1							
Код операции	1001 0100 0010 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Устанавливает в «1» флаг отрицательного значения N регистра SREG. Эквивалентна команде BSET 2							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	**1**	—	—
Маш. циклов	1							
Tiny	Да							
Пример	`add r2,r19 ;Сложить r2 и r19` `sen ;Установить флаг отрицательного результата`							

SER Rd

Установка РОН

Операция	Rd = $FF							
Код операции	1110 1111 dddd 1111						1 слово (2 байта)	
Операнды	16 ≤ d ≤ 31							
Описание	Устанавливает все разряды регистра общего назначения в «1». Команда применима только к регистрам из старшей половины регистрового файла							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1							
Tiny	Да							

Пример	Организация цикла с заданным числом повторений
	`clr r16 ;Очистить регистр r16`
	`ser r17 ;Установить регистр r17`
	`out $18,r16 ;Записать в порт B нули`
	`nop`
	`out $18,r17 ;Записать в порт B единицы`							

SES

Установка флага знака

Операция	S = 1							
Код операции	1001 0100 0100 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Устанавливает в «1» флаг знака S регистра SREG. Эквивалентна команде BSET 4							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	**1**	—	—	—	—
Маш. циклов	1							
Tiny	Да							
Пример	`add r2,r19 ;Сложить r2 и r19` `ses ;Установить флаг знака`							

SET

Установка флага T

Операция	T = 1							
Код операции	1001 0100 0110 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Устанавливает в «1» флаг T регистра SREG. Эквивалентна команде BSET 6							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	**1**	—	—	—	—	—	—
Маш. циклов	1							
Tiny	Да							
Пример	`set ;Установить флаг T`							

SEV

Установка флага переполнения дополнительного кода

Операция	V = 1							
Код операции	1001 0100 0011 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Устанавливает в «1» флаг переполнения дополнительного кода V регистра SREG. Эквивалентна команде BSET 3							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	**1**	—	—	—
Маш. циклов	1							
Tiny	Да							
Пример	`add r2,r19 ;Сложить r2 и r19` `sev ;Установить флаг переполнения`							

SEZ

Установка флага нуля

Операция	Z = 1							
Код операции	1001 0100 0001 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Устанавливает в «1» флаг нуля Z регистра SREG. Эквивалентна команде BSET 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	**1**	–
Маш. циклов	1							
Tiny	Да							
Пример	`add r2,r19 ;Сложить r2 и r19` `sez ;Установить флаг нуля`							

SLEEP

Перевод микроконтроллера в режим пониженного энергопотребления

Операция	См. описание режимов пониженного энергопотребления в разделе 24.3.							
Код операции	1001 0101 1000 1000						1 слово (2 байта)	
Операнды	Нет операндов							
Описание	Переводит микроконтроллер в режим пониженного энергопотребления. Конкретные действия зависят от модели микроконтроллера и от выбранного режима. Для получения более подробной информации обратитесь к разделу 24.3							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	1							
Tiny	Да							

Пример	`mov r0,r11 ;Скопировать регистр r11 в r0`
	`ldi r16,(1<<SE) ;Разрешить переход в «спящий» режим`
	`out MCUCR,r16 ;`
	`sleep ;Переключиться в «спящий» режим`							

SPM

Изменение содержимого памяти программ

<table>
<tr><td>Операция</td><td colspan="8">Зависит от контекста:
{Z} = $FFFF — стирание страницы памяти программ;
{Z} = R1:R0 — запись слова в память программ или запись страницы в буфер;
{Z} = TEMP — пересылка содержимого буфера страницы в память программ;
{BLBITS} = R1:R0 — запись ячеек защиты загрузчика</td></tr>
<tr><td>Код операции</td><td colspan="6">1001 0101 1110 1000</td><td colspan="2">1 слово (2 байта)</td></tr>
<tr><td>Операнды</td><td colspan="8">Нет операндов</td></tr>
<tr><td>Описание</td><td colspan="8">Эта команда может выполнять стирание отдельной страницы памяти программ, запись страницы памяти программ, а также изменение ячеек защиты загрузчика. Адрес страницы памяти программ или слова в ней содержится в регистре Z, а данные, если они необходимы, находятся в регистровой паре R1:R0.
Подробно использование этой команды описано в четвертой части книги</td></tr>
<tr><td rowspan="2">Регистр SREG</td><td>I</td><td>T</td><td>H</td><td>S</td><td>V</td><td>N</td><td>Z</td><td>C</td></tr>
<tr><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr>
<tr><td>Маш. циклов</td><td colspan="8">Зависит от операции</td></tr>
<tr><td>Tiny</td><td colspan="8">Нет</td></tr>
<tr><td>Пример</td><td colspan="8">Запись слова в память программ
<pre>
ldi r31,$F0   ;
clr r30   ;Загрузить адрес в регистр Z
ldi r16,$CF   ;Загрузить данные
mov r1,r16
ldi r16,$FF
mov r0,r16
ldi r16,$03   ;Разрешить SPM (стереть страницу)
out SPMCR,r16 ;
spm   ;Стереть страницу памяти программ по адресу
               ;$F000
ldi r16,$01   ;Разрешить SPM (записать слово в память
               ;программ)
out SPMCR,r16 ;
spm   ;Записать содержимое r1:r0 по адресу $F000
</pre></td></tr>
</table>

ST X, Rr

Косвенная запись в память данных

Операция	[X] = Rr							
Код операции	1001 001r rrrr 1100						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31							
Описание	Сохраняет содержимое регистра общего назначения Rr в памяти данных. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре X							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r27 ;Очистить старший байт индексного регистра`
	`ldi r26,$60 ;Загрузить младший байт адреса`
	`st X,r1 ;Загрузить r1 по адресу $0060`							

ST X+, Rr

Косвенная запись в память данных с постинкрементом

Операция	[X] = Rr, X = X + 1							
Код операции	1001 001r rrrr 1101						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31, для r = 26 или 27 результат операции не определен							
Описание	Сохраняет содержимое регистра общего назначения Rr в памяти данных. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре X. После пересылки байта содержимое регистра X увеличивается на 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r27 ;Очистить старший байт индексного регистра`
	`ldi r26,$60 ;Загрузить младший байт адреса`
	`st X+,r1 ;Загрузить r1 по адресу $0060`
	`in r1,r26 ;В регистре r1 - $61`							

ST –X, Rr

Косвенная запись в память данных с преддекрементом

Операция	X = X – 1, [X] = Rr							
Код операции	1001 001r rrrr 1110						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31, для r = 26 или 27 результат операции не определен							
Описание	Сохраняет содержимое регистра общего назначения Rr в памяти данных. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре X, причем перед обращением к памяти данных содержимое регистра X уменьшается на 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r27 ;Очистить старший байт индексного регистра`
	`ldi r26,$63 ;Загрузить младший байт адреса`
	`st -X,r3 ;Загрузить r3 по адресу $0062`							

ST Y, Rr

Косвенная запись в память данных

Операция	[Y] = Rr							
Код операции	1000 001r rrrr 1000						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31							
Описание	Сохраняет содержимое регистра общего назначения Rr в памяти данных. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Y							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r29 ;Очистить старший байт индексного регистра`
	`ldi r28,$60 ;Загрузить младший байт адреса`
	`st Y,r1 ;Переслать r1 по адресу $0060`							

ST Y+, Rr

Косвенная запись в память данных с постинкрементом

Операция	[Y] = Rr, Y = Y + 1							
Код операции	1001 001r rrrr 1101						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31, для r = 28 или 29 результат операции не определен							
Описание	Сохраняет содержимое регистра общего назначения Rr в памяти данных. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Y. После пересылки байта содержимое регистра Y увеличивается на 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r29 ;Очистить старший байт индексного регистра`
	`ldi r28,$60 ;Загрузить младший байт адреса`
	`st Y+,r1 ;Переслать r1 по адресу $0060`
	`in r2,r28 ;В регистре r2 - $61`							

ST –Y, Rr

Косвенная запись в память данных с преддекрементом

Операция	Y = Y – 1, [Y] = Rr							
Код операции	1001 001r rrrr 1010						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31, для r = 28 или 29 результат операции не определен							
Описание	Сохраняет содержимое регистра общего назначения Rr в памяти данных. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Y, причем перед обращением к памяти данных содержимое регистра Y уменьшается на 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r29 ;Очистить старший байт индексного регистра`
	`ldi r28,$63 ;Загрузить младший байт адреса`
	`st -Y,r3 ;Переслать r3 по адресу $0062`							

ST Z, Rr

Косвенная запись в память данных

Операция	[Z] = Rr							
Код операции	1000 001r rrrr 0000						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31							
Описание	Сохраняет содержимое регистра общего назначения Rr в памяти данных. Для модели Tiny в адресное пространство памяти данных входит только регистровый файл, для остальных — регистровый файл, память ввода/вывода и ОЗУ. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Z							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	2							
Tiny	Да							

Пример	`clr r31 ;Очистить старший байт индексного регистра`
	`ldi r30,$60 ;Загрузить младший байт адреса`
	`st Z,r1 ;Переслать r1 по адресу $0060`							

ST Z+, Rr

Косвенная запись в память данных с постинкрементом

Операция	[Z] = Rr, Z = Z + 1							
Код операции	1001 001r rrrr 1001						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31, для r = 30 или 31 результат операции не определен							
Описание	Сохраняет содержимое регистра общего назначения Rr в памяти данных. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Z. После пересылки байта содержимое регистра Z увеличивается на 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r31 ;Очистить старший байт индексного регистра`
	`ldi r30,$60 ;Загрузить младший байт адреса`
	`st Z+,r1 ;Переслать r1 по адресу $0060`
	`in r2,r30 ;В регистре r2 - $61`							

ST –Z, Rr

Косвенная запись в память данных с преддекрементом

Операция	Z = Z – 1, [Z] = Rr							
Код операции	1001 001r rrrr 1010						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31, для r = 30 или 31 результат операции не определен							
Описание	Сохраняет содержимое регистра общего назначения Rr в памяти данных. Адрес ячейки памяти, к которой производится обращение, содержится в индексном регистре Z, причем перед обращением к памяти данных содержимое регистра Z уменьшается на 1							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r31 ;Очистить старший байт индексного регистра`
	`ldi r30,$63 ;Загрузить младший байт адреса`
	`st -Z,r3 ;Переслать r3 по адресу $0062`							

STD Y+q, Rr

Косвенная относительная запись в память данных

Операция	[Y + q] = Rr							
Код операции	10q0 qq1r rrrr 1qqq						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31, 0 ≤ q ≤ 63							
Описание	Сохраняет содержимое регистра общего назначения Rr в памяти данных. Адрес ячейки памяти, к которой производится обращение, получается суммированием содержимого индексного регистра Y и константы q. Содержимое индексного регистра не изменяется							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r29 ;Очистить старший байт индексного регистра`
	`ldi r28,$63 ;Загрузить младший байт адреса`
	`std Y+2,r4 ;Переслать r4 по адресу $0065`							

STD Z+q, Rd

Косвенная относительная запись в память данных

Операция	[Z + q] = Rr							
Код операции	10q0 qq1r rrrr 0qqq						1 слово (2 байта)	
Операнды	0 ≤ r ≤ 31, 0 ≤ q ≤ 63							
Описание	Сохраняет содержимое регистра общего назначения Rr в памяти данных. Адрес ячейки памяти, к которой производится обращение, получается суммированием содержимого индексного регистра Z и константы q. Содержимое индексного регистра не изменяется							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`clr r31 ;Очистить старший байт индексного регистра`
	`ldi r30,$63 ;Загрузить младший байт адреса`
	`std Z+2,r4 ;Переслать r4 по адресу $0065`							

STS k, Rd

Непосредственная запись в память данных

Операция	[k] = Rd							
Код операции	1001 001d dddd 0000 kkkk kkkk kkkk kkkk						2 слова (4 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ k ≤ 65535							
Описание	Сохраняет содержимое регистра общего назначения Rd в памяти данных. Адрес ячейки памяти, к которой производится обращение, задается константой K							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	2							
Tiny	Нет							

Пример	`lds r2,$FF00 ;r2 = [$FF00]`
	`add r2,r1 ;r2 = r2 + r1`
	`sts $FF00,r2 ;Записать результат по тому же адресу`							

SUB Rd, Rr

Вычитание двух РОН

Операция	Rd = Rd – Rr							
Код операции	0001 10rd dddd rrrr						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31, 0 ≤ r ≤ 31							
Описание	Вычитает из регистра Rd содержимое регистра Rr. Результат помещается в регистр Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	⇔	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							

Пример	`sub r13,r12 ;Вычесть r12 из r13 (r13 = r13 - r12)`
	`brne noteq ;Перейти, если r12 ≠ r13`
	`...`
	`noteq:`
	`...`							

SUBI Rd, K

Вычитание константы из регистра

Операция	Rd = Rd – K							
Код операции	1010 KKKK dddd KKKK						1 слово (2 байта)	
Операнды	16 ≤ d ≤ 31, K = 0...255							
Описание	Вычитает из регистра Rd значение константы K. Результат помещается обратно в регистр. Данная команда применима только к старшей половине регистров общего назначения							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	⇔	⇔	⇔	⇔	⇔	⇔
Маш. циклов	1							
Tiny	Да							

Пример	`subi r22,$11 ;Вычесть $11 из r22`
	`brne noteq ;Перейти, если r22 ≠ $11`
	`...`
	`noteq:`
	`...`							

SWAP Rd

Перестановка тетрад РОН

Операция	Rd(7:4) = Rd(3:0), Rd(3:0) = Rd(7:4)							
Код операции	1001 010d dddd 0010						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Команда производит перестановку старшего и младшего полубайта содержимого регистра Rd							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	—	—	—	—	—
Маш. циклов	1							
Tiny	Да							

```
Пример  inc r1          ;Инкрементировать r1
        swap r1         ;Переставить тетрады
        inc r1          ;Инкрементировать старший полубайт r1
        swap r1         ;Переставить тетрады обратно
```

TST Rd

Проверка РОН на нулевое или отрицательное значение

Операция	Rd = Rd • Rd							
Код операции	0010 00dd dddd dddd						1 слово (2 байта)	
Операнды	0 ≤ d ≤ 31							
Описание	Проверяет содержимое регистра на нулевое или отрицательное значение, путем выполнения операции «Логическое И» регистра с самим собой. Содержимое регистра Rd не изменяется							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	—	—	—	⇔	**0**	⇔	⇔	—
Маш. циклов	1							
Tiny	Да							

```
Пример  tst r0          ;Проверить r0
        breq zero       ;Перейти, если r0 = 0
        ...
        zero:
        ...
```

WDR

Сброс сторожевого таймера

Операция	Рестарт сторожевого таймера							
Код операции	1001 0101 1010 1000						1 слово (2 байта)	
Операнды	Нет							
Описание	Выполняет сброс сторожевого таймера. При включенном сторожевом таймере данная команда должна выполняться через определенный промежуток времени, определяемый коэффициентом деления предделителя сторожевого таймера. Для получения подробной информации обратитесь к Главе 6							
Регистр SREG	**I**	**T**	**H**	**S**	**V**	**N**	**Z**	**C**
	–	–	–	–	–	–	–	–
Маш. циклов	1							
Tiny	Да							
Пример	`wdr ;Сбросить сторожевой таймер`							

Часть 4
Программирование микроконтроллеров семейств Tiny и Mega

Глава 21. **Введение в программирование микроконтроллеров AVR**

21.1. Общие сведения

В общей сложности микроконтроллеры семейств Tiny и Mega поддерживают следующие режимы программирования:

- последовательное программирование при высоком напряжении;
- последовательное программирование при низком напряжении (по интерфейсу SPI);
- параллельное программирование при высоком напряжении;
- программирование по интерфейсу JTAG.

Под «высоким» напряжением здесь понимается управляющее напряжение (12 В), подаваемое на вывод RESET микроконтроллера для перевода последнего в режим программирования.

Какие из режимов поддерживает конкретный микроконтроллер можно узнать, обратившись к **Табл. 4.1**.

Таблица 4.1. **Режимы программирования микроконтроллеров семейств Tiny и Mega**

№	Режим программирования	ATtiny11x	ATtiny12x	ATtiny15L	ATtiny28x	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega323x	ATmega64x	ATmega128x
1	Последовательное, при высоком напряжении	♦	♦	♦	—	—	—	—	—	—	—	—	—	—
2	Последовательное, по интерфейсу SPI	—	♦	♦	—	♦	♦	♦	♦	♦	♦	♦	♦	♦
3	Параллельное, при высоком напряжении	—	—	—	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦
4	По интерфейсу JTAG	—	—	—	—	—	—	♦	—	♦	—	♦	♦	♦

Микроконтроллеры семейства Mega, кроме того, имеют возможность самопрограммирования. Под этим термином понимается изменение содержимого памяти программ, управляемое самим микроконтроллером.

В процессе программирования могут выполняться следующие операции:

- стирание кристалла (Chip erase);
- чтение/запись FLASH-памяти программ;
- чтение/запись EEPROM-памяти данных;
- чтение/запись конфигурационных ячеек;
- чтение/запись ячеек защиты;
- чтение ячеек идентификатора;
- чтение калибровочного байта.

Все модели микроконтроллеров поставляются со стертой памятью программ и памятью данных (во всех ячейках находится число «$FF») и пригодны к немедленному программированию.

21.2. Защита кода и данных

Содержимое FLASH-памяти (память программ), а также содержимое EEPROM-памяти (память данных) может быть защищено от записи и/или чтения посредством программирования ячеек защиты (Lock Bits) LB1 и LB2. Возможные режимы защиты, соответствующие различным состояниям этих ячеек, приведены в **Табл. 4.2**.

Таблица 4.2. **Режимы защиты**

Ячейки защиты			Описание
№ режима	LB1	LB2	
1	1	1	Защита кода и данных отключена
2	0	1	Последующая запись FLASH и EEPROM запрещена
3	0	0	Запрещены запись и чтение FLASH и EEPROM

В режимах 2 и 3 запрещается также изменение конфигурационных ячеек (см. далее). Поэтому включение защиты следует выполнять в самую последнюю очередь, после программирования остальных областей памяти микроконтроллера.

Микроконтроллеры семейства Mega имеют четыре дополнительные ячейки защиты BLB02, BLB01, BLB12 и BLB11. Ячейки BLB02:BLB01 определяют уровень доступа из секции загрузчика к коду, расположенному в секции прикладной программы, а ячейки

BLB12:BLB11 — наоборот, определяют уровень доступа из секции прикладной программы к коду, расположенному в секции загрузчика. Возможные режимы защиты, соответствующие различным состояниям этих ячеек, приведены в **Табл. 4.3** и **Табл. 4.4** соответственно.

Таблица 4.3. **Режимы защиты секции прикладной программы**

Ячейки защиты			Описание
№ режима	BLB02	BLB01	
1	1	1	Нет никаких ограничений по доступу к коду, расположенному в секции прикладной программы
2	1	0	Команда SPM не может осуществлять запись по адресам, находящимся в пределах секции прикладной программы
3	0	0	Команда SPM не может осуществлять запись по адресам, находящимся в пределах секции прикладной программы и команда LPM (ELPM), вызываемая из секции загрузчика, не может осуществлять чтение из секции прикладной программы. Если таблица векторов прерываний расположена в секции загрузчика, прерывания запрещены при вызове их из секции прикладной программы
4	0	1	Команда LPM (ELPM), вызываемая из секции загрузчика, не может осуществлять чтение из секции прикладной программы. Если таблица векторов прерываний расположена в секции загрузчика, прерывания запрещены при вызове их из секции прикладной программы

Таблица 4.4. **Режимы защиты секции загрузчика**

Ячейки защиты			Описание
№ режима	BLB12	BLB11	
1	1	1	Нет никаких ограничений по доступу к коду, расположенному в секции загрузчика
2	1	0	Команда SPM не может осуществлять запись по адресам, находящимся в пределах секции загрузчика
3	0	0	Команда SPM не может осуществлять запись по адресам, находящимся в пределах секции загрузчика и команда LPM (ELPM), вызываемая из секции прикладной программы, не может осуществлять чтение из секции загрузчика. Если таблица векторов прерываний расположена в секции прикладной программы, прерывания запрещены при вызове их из секции загрузчика
4	0	1	Команда LPM (ELPM), вызываемая из секции прикладной программы, не может осуществлять чтение из секции загрузчика. Если таблица векторов прерываний расположена в секции прикладной программы, прерывания запрещены при вызове их из секции загрузчика

Все перечисленные ячейки защиты сгруппированы в одном байте. Расположение ячеек защиты в нем для разных семейств приведено на **Рис. 4.1**.

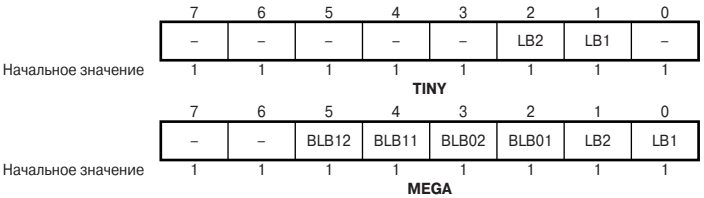


Рис. 4.1. Байт ячеек защиты

В исходном (незапрограммированном) состоянии во всех ячейках защиты содержится «1», после программирования — «0». Стирание ячеек (запись в них лог. 1) можно произвести только при выполнении команды «Стирание кристалла», уничтожающей также содержимое FLASH- и EEPROM-памяти.

21.3. Конфигурационные ячейки

Как следует из названия, конфигурационные ячейки (Fuse Bits) определяют некоторые параметры конфигурации микроконтроллера. Эти ячейки расположены в отдельном адресном пространстве, доступном только при программировании. Все конфигурационные ячейки сгруппированы в несколько байтов (от одного до трех в зависимости от модели), а состав этих ячеек зависит от конкретной модели микроконтроллера. Наличие тех или иных ячеек в конкретном микроконтроллере можно определить по **Табл. 4.5** (семейство Tiny) и **Табл. 4.6** (семейство Mega), где в столбцах, отмеченных «звездочкой», указаны состояния конфигурационных ячеек по умолчанию.

Таблица 4.5. **Конфигурационные ячейки микроконтроллеров семейства Tiny**

Разряд	ATtiny11x		ATtiny12x		ATtiny15L		ATtiny28x	
	Название	*	Название	*	Название	*	Название	*
7	—	1	BODLEVEL	0	BODLEVEL	0	—	1
6	—	1	BODEN	1	BODEN	1	—	1
5	—	1	SPIEN	0	SPIEN	0	—	1
4	FSTRT	1	RSTDISBL	1	RSTDISBL	1	INTCAP	1
3	RSTDISBL	1	CKSEL3	0	—	1	CKSEL3	0
2	CKSEL2	1	CKSEL2	0	—	1	CKSEL2	0
1	CKSEL1	0	CKSEL1	1	CKSEL1	0	CKSEL1	1
0	CKSEL0	0	CKSEL0	0	CKSEL0	0	CKSEL0	0

Таблица 4.6. **Название и состояние (*) конфигурационных ячеек микроконтроллеров семейства Mega**

Разряд	ATmega8x		ATmega8515x		ATmega16x		ATmega161x		ATmega162x		ATmega163x		ATmega323x		ATmega64x ATmega128x	
	Название	*	Название	*	Название	*	Название	*	Название	*	Название	*	Название	*	Название	*
Младший конфигурационный байт																
7	BODLEVEL	1	BODLEVEL	1	BODLEVEL	1	—	1	CDIV16	0	BODLEVEL	1	BODLEVEL	1	BODLEVEL	1
6	BODEN	1	BODEN	1	BODEN	1	BOOTRST	1	CKOUT	1	BODEN	1	BODEN	1	BODEN	1
5	SUT1	1	SUT1	1	SUT1	1	SPIEN	0	SUT1	1	SPIEN	0	—	1	SUT1	1
4	SUT0	0	SUT0	0	SUT0	0	BODLEVEL	1	SUT0	0	—	1	—	1	SUT0	0
3	CKSEL3	0	CKSEL3	0	CKSEL3	0	BODEN	1	CKSEL3	0	CKSEL3	0	CKSEL3	0	CKSEL3	0
2	CKSEL2	0	CKSEL2	0	CKSEL2	0	CKSEL2	0	CKSEL2	0	CKSEL2	0	CKSEL2	0	CKSEL2	0
1	CKSEL1	0	CKSEL1	0	CKSEL1	0	CKSEL1	1	CKSEL1	1	CKSEL1	1	CKSEL1	1	CKSEL1	0
0	CKSEL0	1	CKSEL0	1	CKSEL0	1	CKSEL0	0	CKSEL0	0	CKSEL0	0	CKSEL0	0	CKSEL0	1
Старший конфигурационный байт																
7	RSTDISBL	1	S8515C	1	OCDEN	1	—	—	OCDEN	1	—	1	OCDEN	1	OCDEN	1
6	WDTON	1	WDTON	1	JTAGEN	0	—	—	JTAGEN	0	—	1	JTAGEN	0	JTAGEN	0
5	SPIEN	0	SPIEN	0	SPIEN	0	—	—	SPIEN	0	—	1	SPIEN	0	SPIEN	0
4	CKPOT	1	CKPOT	1	CKPOT	1	—	—	WDTON	1	—	1	—	1	CKPOT	1
3	EESAVE	1	EESAVE	1	EESAVE	1	—	—	EESAVE	1	—	1	EESAVE	1	EESAVE	1
2	BOOTSZ1	0	BOOTSZ1	0	BOOTSZ1	0	—	—	BOOTSZ1	0	BOOTSZ1	1	BOOTSZ1	1	BOOTSZ1	0
1	BOOTSZ0	0	BOOTSZ0	0	BOOTSZ0	0	—	—	BOOTSZ0	0	BOOTSZ0	1	BOOTSZ0	1	BOOTSZ0	0
0	BOOTRST	1	BOOTRST	1	BOOTRST	1	—	—	BOOTRST	1	BOOTRST	1	BOOTRST	1	BOOTRST	1
Дополнительный конфигурационный байт																
7	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	1
6	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	1
5	—	—	—	—	—	—	—	—	—	1	—	—	—	—	—	1
4	—	—	—	—	—	—	—	—	M161C	1	—	—	—	—	—	1
3	—	—	—	—	—	—	—	—	BODLEVEL2	1	—	—	—	—	—	1
2	—	—	—	—	—	—	—	—	BODLEVEL1	1	—	—	—	—	—	1
1	—	—	—	—	—	—	—	—	BODLEVEL0	1	—	—	—	—	M103C	0
0	—	—	—	—	—	—	—	—	—	1	—	—	—	—	WDTON	1

Краткое назначение всех конфигурационных ячеек приведено в **Табл. 4.7.** Подробное описание их назначений было приведено в соответствующих главах книги.

Таблица 4.7. **Назначение конфигурационных ячеек**

Название	ATtiny11x	ATtiny12x	ATtiny15L	ATtiny28x	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega323x	ATmega64x	ATmega128x	Назначение
FSTRT	♦	—	—	—	—	—	—	—	—	—	—	—	—	Определяет длительность задержки сброса t_{TOUT}
RSTDISBL	♦	♦	♦	♦	♦	—	—	—	—	—	—	—	—	Определяет функционирование вывода микроконтроллера, совмещенного с выводом аппаратного сброса (0 — контакт порта ввода/вывода, 1 — вывод сброса)
CKSEL	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	♦	Определяют режим работы тактового генератора, а также длительность задержки сброса t_{TOUT}
BODLEVEL	—	♦	♦	—	♦	♦	♦	♦	♦	♦	♦	♦	♦	Определяет порог срабатывания схемы BOD
BODEN	—	♦	♦	—	♦	♦	♦	♦	—	♦	♦	♦	♦	Разрешает/запрещает функционирование схемы BOD (0 — разрешено, 1 — запрещено)
SPIEN*	—	♦	♦	—	♦	♦	♦	♦	♦	♦	♦	♦	♦	Разрешает/запрещает программирование по интерфейсу SPI (0 — разрешено, 1 — запрещено)
INTCAP	—	—	—	♦	—	—	—	—	—	—	—	—	—	Определяет состояние внутренних конденсаторов, подключаемых между выводами XTAL1, XTAL2 и общим проводом (0 — подключены, 1 — отключены)
SUT	—	—	—	—	♦	♦	♦	—	♦	—	—	♦	♦	Определяет длительность задержки сброса t_{TOUT}
WDTON	—	—	—	—	♦	♦	—	—	♦	—	—	♦	♦	Определяет режим работы сторожевого таймера (0 — всегда включен, 1 — может быть выключен программно)
CKPOT	—	—	—	—	♦	♦	♦	—	—	—	—	♦	♦	Определяет функционирование тактового генератора, действие зависит от установок ячеек CKSEL

Продолжение таблицы 4.7

Название	ATtiny11x	ATtiny12x	ATtiny15L	ATtiny28x	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega323x	ATmega64x	ATmega128x	Назначение
EESAVE**	—	—	—	—	♦	♦	♦	—	♦	—	♦	♦	♦	Определяет влияние команды «Стирание кристалла» на EEPROM-память (0 — не стирает, 1 — стирает)
BOOTSZ	—	—	—	—	♦	♦	♦	—	♦	♦	♦	♦	♦	Определяют размер секции загрузчика
BOOTRST	—	—	—	—	♦	♦	♦	—	♦	♦	♦	♦	♦	Определяет положение вектора сброса
S8515C	—	—	—	—	—	♦	—	—	—	—	—	—	—	Включает/выключает режим совместимости с микроконтроллерами AT90S4414/8515 семейства Classic (0 — включен, 1 — выключен)
OCDEN	—	—	—	—	—	—	♦	—	♦	—	♦	♦	♦	Разрешает/запрещает внутрисхемную отладку (0 — разрешена, 1 — запрещена)
JTAGEN	—	—	—	—	—	—	♦	—	♦	—	♦	♦	♦	Разрешает/запрещает использование интерфейса JTAG (0 — разрешен, 1 — запрещен)
CDIV16	—	—	—	—	—	—	—	—	♦	—	—	—	—	Определяет начальное состояние делителя системного тактового сигнала
CKOUT	—	—	—	—	—	—	—	—	♦	—	—	—	—	Определяет состояние выходного буфера системного тактового сигнала (0 — подключен к выводу микроконтроллера, 1 — отключен)
M161C	—	—	—	—	—	—	—	—	♦	—	—	—	—	Включает/выключает режим совместимости с микроконтроллерами ATmega161x семейства Mega (0 — включен, 1 — выключен)
M103C	—	—	—	—	—	—	—	—	—	—	—	♦	♦	Включает/выключает режим совместимости с микроконтроллерами ATmega103x семейства Mega (0 — включен, 1 — выключен)

* В микроконтроллерах семейства Mega при программировании по последовательному каналу ячейка SPIEN недоступна.

** Изменение состояния этой ячейки вступает в силу сразу же после ее программирования, а у остальных ячеек — после выхода из режима программирования.

Для изменения содержимого конфигурационных ячеек используются специальные команды программирования. Команда «Стирание кристалла» на состояние этих ячеек не влияет. Напоминаем, что при запрограммированной ячейке защиты LB1 конфигурационные ячейки блокируются. Поэтому конфигурацию микроконтроллера необходимо задавать до программирования ячеек защиты.

21.4. Идентификатор

Все микроконтроллеры фирмы «Atmel» имеют три 8-разрядные ячейки, содержимое которых позволяет идентифицировать устройство. Как и конфигурационные ячейки, ячейки идентификатора расположены в отдельном адресном пространстве, доступ к которому возможен только в режиме программирования. Однако ячейки идентификатора в отличие от конфигурационных ячеек по понятным причинам доступны только для чтения. Содержимое ячеек идентификатора для всех микроконтроллеров рассматриваемых семейств приведено в **Табл. 4.8.**

Таблица 4.8. **Ячейки идентификатора**

Адрес	ATtiny11x	ATtiny12x	ATtiny15L	ATtiny28x	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega323x	ATmega64x	ATmega128x	Описание
$00	$1E	$1E	$1E	$1E	$1E	$1E	$1E	$1E	$1E	$1E	$1E	$1E	$1E	Код производителя («Atmel»)
$01	$90	$90	$90	$91	$93	$93	$94	$94	$94	$94	$95	$96	$97	Код объема FLASH-памяти*
$02	$04	$05	$06	$07	$07	$06	$03	$01	$04	$02	$01	$02	$02	Код устройства
* $90 — 1 Кбайт, $91 — 2 Кбайт, $93 — 8 Кбайт, $94 — 16 Кбайт, $95 — 32 Кбайт, $96 — 64 Кбайт, $97 — 128 Кбайт.														

Как видно из таблицы, значение кода устройства (ячейка $02) может совпадать для различных моделей. Поэтому устройство можно идентифицировать только по совокупности значений ячеек $01 и $02, т. к. именно эта пара чисел является уникальной для каждого микроконтроллера.

Обратите внимание на то, что у микроконтроллеров ATtiny11x/12x в режиме защиты 3 (обе ячейки защиты запрограммированы),

идентификатор доступен только в режиме программирования при высоком напряжении. При чтении ячеек $00, $01 и $02 идентификатора в режиме низковольтного программирования возвращаются значения $00, $01 и $02 соответственно.

21.5. Калибровочная ячейка

В калибровочную ячейку при изготовлении микроконтроллера заносится калибровочная константа, предназначенная для подстройки на номинальную частоту внутреннего *RC*-генератора. Количество этих ячеек зависит от того, на скольких частотах может работать внутренний *RC*-генератор. В моделях ATtiny12x/15L/28x и ATmega162x/163x/323x имеется одна, а в моделях ATmega8x/8515x/16x/64x/128x — четыре 8-разрядных ячейки. Располагаются они в старших байтах адресного пространства ячеек идентификатора (одна ячейка — по адресу $000, четыре ячейки — по адресам $000, $001, $002 и $003 для частот 1, 2, 4 и 8 МГц соответственно). В моделях ATtiny11x и ATmega161x калибровочная ячейка отсутствует.

Использование этих ячеек зависит от модели микроконтроллера. В микроконтроллерах семейства Tiny программатор должен прочитать содержимое калибровочной ячейки и занести его по какому-либо адресу FLASH-памяти программ. А программа должна после старта считать это значение из памяти программ и занести его в регистр OSCCAL. Сказанное справедливо также для микроконтроллеров ATmega163x и ATmega323x семейства Mega.

В остальных моделях занесение калибровочной константы в регистр OSCCAL осуществляется аппаратно при нахождении микроконтроллера в состоянии сброса. Однако в моделях ATmega8x/8515x/16x/64x/128x генератор автоматически калибруется только на одну частоту — 1 МГц. Поэтому при использовании другого режима работы *RC*-генератора его калибровку необходимо осуществлять вручную, как описано в предыдущем абзаце.

21.6. Организация памяти программ и данных микроконтроллеров семейства Mega

В микроконтроллерах семейства Mega используется страничная организация памяти программ. При программировании весь объем FLASH-памяти разбивается на т. н. «страницы», размеры и количество которых зависят от конкретной модели микроконтроллера (**Табл. 4.9**).

Таблица 4.9. **Параметры страничной организации памяти программ**

Параметр	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega323x	ATmega64x	ATmega128x
Размер памяти программ (в 16-разрядных словах) [байт]	4K	4K	8K	8K	8K	8K	16K	32K	64K
Количество слов на странице	32	32	64	64	64	64	64	128	128
Количество страниц	128	128	128	128	128	128	256	256	512

Соответственно, при программировании памяти программ микроконтроллеров семейства Mega данные сначала загружаются в буфер страницы и только затем заносятся непосредственно в память программ. Прошивка всех ячеек страницы при этом осуществляется одновременно.

Таким же образом организована и EEPROM-память следующих моделей: ATmega8x, ATmega8515x, ATmega16x, ATmega162x, ATmega64x и ATmega128x. Размер страниц EEPROM-памяти указанных микроконтроллеров, а также их количество приведены в **Табл. 4.10**.

Таблица 4.10. **Параметры страничной организации EEPROM-памяти**

Параметр	ATmega8x	ATmega8515x	ATmega16x	ATmega162x	ATmega64x	ATmega128x
Размер EEPROM-памяти [байт]	512	512	512	512	2K	4K
Размер страницы [байт]	4	4	4	4	8	8
Количество страниц	128	128	128	128	256	512

Однако следует отметить, что в большинстве этих моделей страничная организация EEPROM-памяти используется только при программировании ее в параллельном режиме. Программирование этой области памяти по последовательному каналу осуществляется так же как и в остальных моделях, т. е. побайтно.

Глава 22. Последовательное программирование при высоком напряжении

22.1. Общие сведения

Режим последовательного программирования при высоком напряжении поддерживается не всеми моделями, а только ATtiny11x/12x/15L. Этот режим требует дополнительного источника повышенного напряжения (12 В) и применяется, как правило, для программирования микроконтроллеров перед установкой их на плату. Схема включения микросхем в этом режиме приведена на **Рис. 4.2.** Временные диаграммы сигналов микроконтроллера представлены на **Рис. 4.3**, а значения параметров сигналов приведены в **Табл. 4.11** и **Табл. 4.12**.

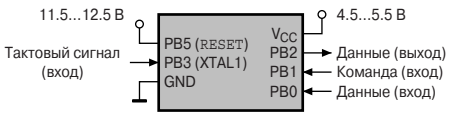


Рис. 4.2. ATtiny11x/12x/15L в режиме программирования

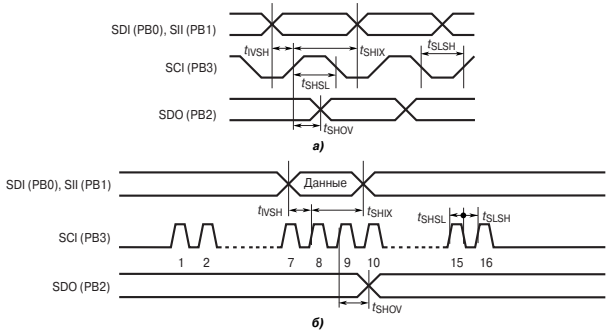


Рис. 4.3. Временные диаграммы сигналов в режиме последовательного программирования для ATtiny11x/12x (*а*) и ATtiny15L (*б*)

Таблица 4.11. **Параметры сигналов в режиме последовательного программирования при высоком напряжении для ATtiny11x/12x**

Обозначение	Параметр	min	typ	max
t_{SHSL}	Длительность положительного импульса сигнала SCI (PB3) [нс]	100	—	—
t_{SLSH}	Длительность отрицательного импульса сигнала SCI (PB3) [нс]	100	—	—
t_{IVSH}	Задержка переднего фронта сигнала SCI (PB3) относительно момента установления сигналов SDI (PB0) и SII (PB1) [нс]	50	—	—
t_{SHIX}	Время удержания сигналов SDI (PB0) и SII (PB1) относительно переднего фронта сигнала SCI (PB3) [нс]	50	—	—
t_{SHOV}	Задержка момента установления сигнала SDO (PB2) относительно переднего фронта сигнала SCI (PB3) [нс]	10	16	32
t_{WLWH_PFB}	Период ожидания после 3-й посылки команды «Запись конфигурационных ячеек» [мс]	1.7	2.5	3.4

Таблица 4.12. **Параметры сигналов в режиме последовательного программирования при высоком напряжении для ATtiny15L**

Обозначение	Параметр	min	typ	max
t_{SHSL}	Длительность положительного импульса сигнала SCI (PB3) [нс]	25	—	—
t_{SLSH}	Длительность отрицательного импульса сигнала SCI (PB3) [нс]	25	—	—
t_{IVSH}	Задержка переднего фронта 8-го импульса сигнала SCI (PB3) относительно момента установления сигналов SDI (PB0) и SII (PB1) [нс]	50	—	—
t_{SHIX}	Время удержания сигналов SDI (PB0) и SII (PB1) относительно переднего фронта 8-го импульса сигнала SCI (PB3) [нс]	50	—	—
t_{SHOV}	Задержка момента установления сигнала SDO (PB2) относительно переднего фронта 9-го импульса сигнала SCI (PB3) [нс]	10	16	32

22.2. Управление процессом программирования

Для перевода микроконтроллера в режим программирования необходимо выполнить следующие действия:

1. Подать напряжение питания (4.5...5.5 В) на микроконтроллер.
2. Подать на выводы PB3 и PB5 напряжение НИЗКОГО уровня на время не менее 100 нс (для ATtiny15L — 30 мкс).
3. Подать на вывод PB3 ATtiny11x/12x не менее 4-х импульсов длительностью большей или равной 100 нс, а затем напряжение НИЗКОГО уровня и выждать не менее 100 нс (для ATtiny15L

подать только напряжение НИЗКОГО уровня и выждать не менее 100 нс).

4. Подать напряжение 12 В на вывод PB5 и выждать не менее 100 нс перед изменением состояния вывода PB3. Первую команду можно будет посылать не ранее чем через 8 мкс.

Выключение микроконтроллера после его программирования выполняется в следующей последовательности:

1. Подать на вывод PB3 напряжение НИЗКОГО уровня.
2. Подать на вывод PB5 напряжение ВЫСОКОГО уровня.
3. Отключить напряжение питания от микроконтроллера.

Всего в этом режиме имеется 16 команд, каждая из которых содержит от 2-х до 4-х 11-разрядных посылок. Передача команд, а также вывод результатов их выполнения осуществляются от старшего разряда к младшему. В микроконтроллерах ATtiny11x/12x и входные, и выходные данные «защелкиваются» по нарастающему фронту тактового сигнала (**Рис. 4.4, а**). В микроконтроллере ATtiny15L входные данные и выходные данные «защелкиваются» по нарастающему фронту 8-го импульса посылки из 16 импульсов внешнего тактового сигнала. Эти 16 импульсов внешнего тактового сигнала необходимы для генерации одного импульса внутреннего тактового сигнала микроконтроллера (**Рис. 4.4, б**).

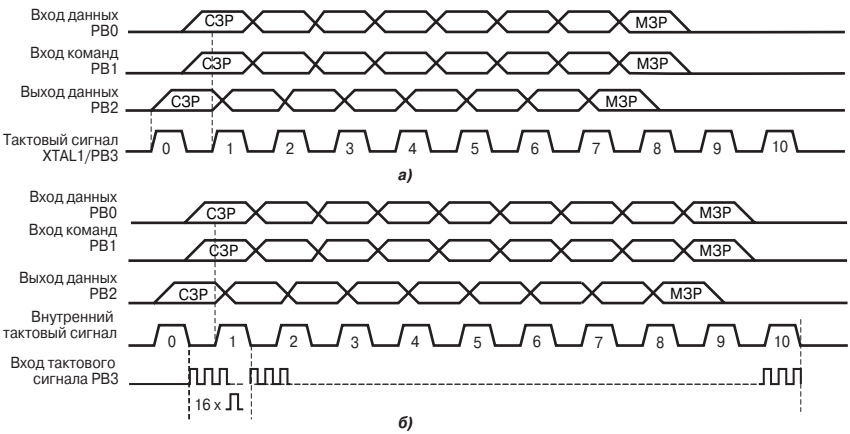


Рис. 4.4. Передача данных в режиме последовательного программирования при высоком напряжении: ATtiny11x/12x (*а*), ATtiny15L (*б*)

Формат всех команд приведен в **Табл. 4.13**. Там же помещены пояснения по применению той или иной команды.

Таблица 4.13. **Команды режима последовательного программирования при высоком напряжении**

Название команды	Вывод	Формат команды				Замечания
		1-я посылка	2-я посылка	3-я посылка	4-я посылка	
Стирание кристалла	PB0	0_1000_0000_00	0_0000_0000_00	0_0000_0000_00	0_0000_0000_00	После 4-й посылки ждать появления на выводе PB2 лог. 1
	PB1	0_0100_1100_00	0_0110_0100_00	0_0110_1100_00	0_0100_1100_00	
	PB2	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	
Загрузка адреса ячейки FLASH-памяти при записи	PB0	0_0001_0000_00	0_0000_000**a**_00	0_**bbbb_bbbb**_00	—	Повторить 2-ю посылку при переходе к новому 256-байтному блоку памяти; повторить 3-ю посылку для каждого нового адреса
	PB1	0_0100_1100_00	0_0001_1100_00	0_0000_1100_00	—	
	PB2	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	—	
Запись младшего байта ячейки FLASH-памяти	PB0	0_**iiii_iiii**_00	0_0000_0000_00	0_0000_0000_00	—	После 3-й посылки ждать появления на выводе PB2 лог. 1; полностью (все три посылки) повторить команду для каждой ячейки
	PB1	0_0010_1100_00	0_0110_0100_00	0_0110_1100_00	—	
	PB2	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	0_0000_0000_00	—	
Запись старшего байта ячейки FLASH-памяти	PB0	0_**iiii_iiii**_00	0_0000_0000_00	0_0000_0000_00	—	После 3-й посылки ждать появления на выводе PB2 лог. 1; полностью (все три посылки) повторить команду для каждой ячейки
	PB1	0_0011_1100_00	0_0111_0100_00	0_0111_1100_00	—	
	PB2	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	0_0000_0000_00	—	
Загрузка адреса ячейки FLASH-памяти при чтении	PB0	0_0001_0010_00	0_0000_000**a**_00	0_**bbbb_bbbb**_00	—	Повторить 2-ю и 3-ю посылки для каждого нового адреса
	PB1	0_0100_1100_00	0_0001_1100_00	0_0000_1100_00	—	
	PB2	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	—	

Продолжение таблицы 4.13

Название команды	Вывод	Формат команды				Замечания
		1-я посылка	2-я посылка	3-я посылка	4-я посылка	
Чтение младшего байта ячейки FLASH-памяти	PB0	0_0000_0000_00	0_0000_0000_00	—	—	Полностью (обе посылки) повторить команду для каждой ячейки
	PB1	0_0110_1000_00	0_0110_1100_00	—	—	
	PB2	x_xxxx_xxxx_xx	**o_oooo_ooo**x_xx	—	—	
Чтение старшего байта ячейки FLASH-памяти	PB0	0_0000_0000_00	0_0000_0000_00	—	—	Полностью (обе посылки) повторить команду для каждой ячейки
	PB1	0_0111_1000_00	0_0111_1100_00	—	—	
	PB2	x_xxxx_xxxx_xx	**o_oooo_ooo**x_xx	—	—	
Загрузка адреса ячейки EEPROM-памяти при записи (ATtiny12x и ATtiny15L)	PB0	0_0001_0001_00	0_00**bb_bbbb**_00	—	—	Повторить 2-ю посылку для каждого нового адреса
	PB1	0_0100_1100_00	0_0000_1100_00	—	—	
	PB2	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	—	—	
Запись ячейки EEPROM-памяти (ATtiny12x и ATtiny15L)	PB0	0_**iiii_iiii**_00	0_0000_0000_00	0_0000_0000_00	—	После 3-й посылки ждать появления на выводе PB2 лог. 1
	PB1	0_0010_1100_00	0_0110_0100_00	0_0110_1100_00	—	
	PB2	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	0_0000_0000_00	—	
Загрузка адреса ячейки EEPROM-памяти при чтении (ATtiny12x и ATtiny15L)	PB0	0_0000_0011_00	0_00**bb_bbbb**_00	—	—	Повторить 2-ю посылку для каждого нового адреса
	PB1	0_0100_1100_00	0_0000_1100_00	—	—	
	PB2	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	—	—	

Продолжение таблицы 4.13

Название команды	Вывод	Формат команды				Замечания
		1-я посылка	2-я посылка	3-я посылка	4-я посылка	
Чтение содержимого ячейки EEPROM-памяти (ATtiny12x и ATtiny15L)	PB0	0_0000_0000_00	0_0000_0000_00	—	—	Повторить 2-ю посылку для каждого нового адреса
	PB1	0_0110_1000_00	0_0110_1100_00	—	—	
	PB2	x_xxxx_xxxx_xx	o_oooo_ooox_xx	—	—	
Запись конфигурационных ячеек	PB0	0_0100_0000_00	0_**FFFF_FFFF**_00	0_0000_0000_00	0_0000_0000_00	ATtiny11x: между 3-й и 4-й посылками требуется выдержать паузу длительностью t_{WLWH_PFB}; ATtiny12x и ATtiny15L: после 4-й посылки ждать появления на выводе PB2 лог. 1
	PB1	0_0100_1100_00	0_0010_1100_00	0_0110_0100_00	0_0110_1100_00	
	PB2	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	
Запись ячеек защиты	PB0	0_0010_0000_00	0_**LLLL_LLLL**_00	0_0000_0000_00	0_0000_0000_00	После 4-й посылки ждать появления на выводе PB2 лог. 1
	PB1	0_0100_1100_00	0_0010_1100_00	0_0110_0100_00	0_0110_1100_00	
	PB2	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	0_0000_0000_00	
Чтение конфигурационных ячеек	PB0	0_0000_0100_00	0_0000_0000_00	0_0000_0000_00	—	—
	PB1	0_0100_1100_00	0_0110_1000_00	0_0110_1100_00	—	
	PB2	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	**F_FFFF_FFF**x_xx	—	

Продолжение таблицы 4.13

Название команды	Вывод	Формат команды				Замечания
		1-я посылка	2-я посылка	3-я посылка	4-я посылка	
Чтение ячеек защиты	PB0	0_0000_0100_00	0_0000_0000_00	0_0000_0000_00	—	—
	PB1	0_0100_1100_00	0_0111_1000_00	0_0111_1100_00	—	
	PB2	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_**LLLL_LLLL**_xx	—	
Чтение ячеек идентификатора	PB0	0_0000_1000_00	0_0000_00**bb**_00	0_0000_0000_00	0_0000_0000_00	Повторить посылки 2...4 для каждой ячейки идентификатора
	PB1	0_0100_1100_00	0_0000_1100_00	0_0110_1000_00	0_0110_1100_00	
	PB2	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	x_xxxx_xxxx_xx	**o**_**oooo**_**ooo**x_xx	
Чтение калибровочной ячейки (ATtiny12x и ATtiny15L)	PB0	0_0000_1000_00	0_0000_ 0000_00	0_0000_0000_00	0_0000_ 0000_00	—
	PB1	0_0100_1100_00	0_0000_ 1100_00	0_0111_1000_00	0_0111_ 1100_00	
	PB2	x_xxxx_xxxx_xx	x_xxxx_ xxxx_xx	x_xxxx_xxxx_xx	**o**_**oooo**_ **ooo**x_xx	

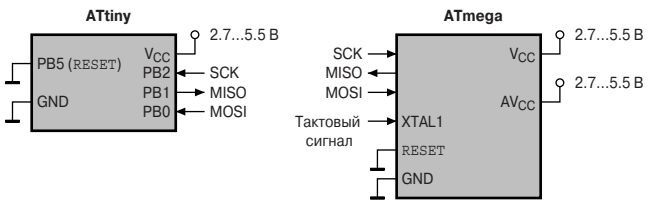
Примечание. Расшифровка условных обозначений, используемых в таблице:
a — разряды старшего байта адреса;
b — разряды младшего байта адреса;
i — посылаемые в микроконтроллер данные;
o — считываемые из микроконтроллера данные;
x — состояние разряда безразлично;
L — разряды байта ячеек защиты (см. **Рис. 4.1**);
F — разряды конфигурационного байта (см. **Табл. 4.5**).

Глава 23. **Программирование по последовательному каналу**

23.1. Общие сведения

Режим программирования по последовательному каналу поддерживается всеми микроконтроллерами семейства Mega, а также микроконтроллерами ATtiny12x и ATtiny15L семейства Tiny. В этом режиме программирование памяти программ и данных осуществляется через последовательный интерфейс SPI. Как правило, рассматриваемый режим используется для программирования (перепрограммирования) микроконтроллера непосредственно в устройстве.

Схема включения микросхем в режиме программирования по последовательному каналу приведена на **Рис. 4.5**.



Замечание (для ATmega) — если в качестве тактового используется внутренний RC-генератор, вывод XTAL1 оставляют неподключенным.

Рис. 4.5. Включение микроконтроллеров в режиме программирования по последовательному каналу

Как видно из рисунка, для подключения программатора к устройству используются три линии интерфейса: SCK (тактовый сигнал), MOSI (вход данных) и MISO (выход данных). Соответствие между линиями интерфейса и контактами портов ввода/вывода всех микроконтроллеров приведено в **Табл. 4.14**.

Таблица 4.14. **Выводы, используемые при программировании по последовательному каналу**

Название линий интерфейса	ATtiny12x	ATtiny15L	ATmega8x	ATmega8515x	ATmega16x	ATmega161x	ATmega162x	ATmega163x	ATmega323x	ATmega64x ATmega128x	Назначение выводов
SCK	PB2	PB2	PB5	PB7	PB7	PB7	PB7	PB7	PB7	PB1	Вход тактового сигнала
MISO (PDO)	PB1	PB1	PB4	PB6	PB6	PB6	PB6	PB6	PB6	PE1	Выход данных
MOSI (PDI)	PB0	PB0	PB3	PB5	PB5	PB5	PB5	PB5	PB5	PE0	Вход данных

Следует иметь в виду, что в моделях ATmega64x и ATmega128x выводы, используемые для программирования, не совпадают с выводами, предназначенными для штатной работы интерфейса SPI.

Временные диаграммы сигналов при программировании микроконтроллеров в рассматриваемом режиме представлены на **Рис. 4.6**, а значения параметров сигналов приведены в **Табл. 4.15**.

Таблица 4.15. **Параметры сигналов при программировании по последовательному каналу**

Обозначение	Параметр		min	typ	max
$1/t_{CLCL}$	Частота тактового сигнала [МГц]	V_{CC} = 2.7...5.5 В (ATtiny15L)	0.8	1.6	—
		V_{CC} = 2.2...2.7 В (ATtiny12V-1)	0	—	1
		V_{CC} = 1.8...3.6 В (ATmega162V)			
		V_{CC} = 2.7...4.0 В (ATtiny12L-4)	0	—	4
		V_{CC} = 2.7...5.5 В (ATmega161L, ATmega163L, ATmega323L)			
		V_{CC} = 4.0...5.5 В (ATtiny12-8, ATmega161, ATmega163, ATmega323)	0	—	8
		V_{CC} = 2.7...5.5 В (ATmega8L, ATmega8515L, ATmega16L, ATmega162L, ATmega64L/128L)			
		V_{CC} = 4.5...5.5 В (ATmega8, ATmega8515, ATmega16, ATmega162, ATmega64/128)	0	—	16
t_{SHSL}	Длительность положительного импульса сигнала SCK [нс]		$2t_{CLCL}$*	—	—
t_{SLSH}	Длительность отрицательного импульса сигнала SCK [нс]		$2t_{CLCL}$*	—	—
t_{OVSH}	Задержка нарастающего фронта сигнала SCK относительно установления сигнала MOSI [нс]		t_{CLCL}	—	—
t_{SHOX}	Время удержания сигнала MOSI относительно нарастающего фронта сигнала SCK [нс]		$2t_{CLC}$*	—	—

Продолжение таблицы 4.15

Обозначение	Параметр		min	typ	max
t_{SLIV}	Задержка установления сигнала MISO относительно спадающего фронта сигнала SCK [нс]		10	16	32
t_{WD_ERASE}	Период ожидания после загрузки команды «Стирание кристалла» [мс]	ATmega161x	28	—	—
		ATmega163x, ATmega323x	32	—	—
		остальные	9	—	—
t_{WD_FLASH}	Период ожидания после записи FLASH-памяти [мс]	ATmega161x	14	—	—
		ATmega163x, ATmega323x	16	—	—
		остальные	4.5	—	—
t_{WD_EEPROM}	Период ожидания после записи EEPROM-памяти [мс]	ATmega161x	3.4	—	—
		ATmega163x ATmega323x	4	—	—
		остальные	9	—	—
t_{WD_FUSE}	Период ожидания после записи конфигурационных ячеек или ячеек защиты [мс]	ATmega163x ATmega323x	2	—	—
* $2t_{CLCL}$ при $f_{CK} < 12$ МГц и $3t_{CLCL}$ при $f_{CK} \geq 12$ МГц.					

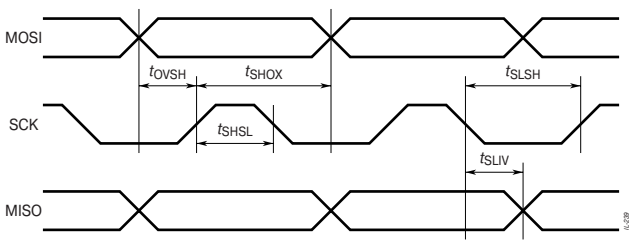


Рис. 4.6. Временные диаграммы сигналов при программировании по последовательному каналу

Как и в рабочем режиме, при программировании по последовательному каналу микроконтроллеру требуется источник тактового сигнала. В качестве такового может использоваться любой из допустимых для микроконтроллера источников. При этом должно выполняться следующее условие: длительность импульсов как НИЗКОГО уровня, так и ВЫСОКОГО уровня сигнала SCK должна быть больше 2-х (при $f_{CK} < 12$ МГц) или 3-х (при $f_{CK} \geq 12$ МГц) периодов тактового сигнала микроконтроллера.

Программирование осуществляется путем посылки 4-байтных команд на вывод MOSI микроконтроллера. Результат выполнения команд чтения снимается с вывода MISO микроконтроллера. Передача команд и выдача результатов их выполнения осуществляется от старшего разряда к младшему. При этом «защелкивание» входных данных выполняется по нарастающему фронту сигнала SCK, а «защелкивание» выходных данных — по спадающему (см. **Рис. 4.6** и **Рис. 4.7**).

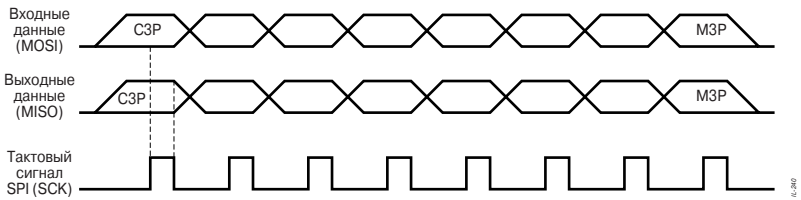


Рис. 4.7. Передача данных при программировании по последовательному каналу

Формат команд, используемых в этом режиме для программирования различных микроконтроллеров семейства, приведен в **Табл. 4.16**.

23.2. Переключение в режим программирования

Переключение микроконтроллера в режим программирования по последовательному каналу выполняется в следующем порядке:

1. Подать на микроконтроллер напряжение питания, при этом на выводах SCK и RESET должно присутствовать напряжение НИЗКОГО уровня. В некоторых случаях (если программатор не гарантирует установку сигнала SCK в «0» при подаче питания) после установки сигнала SCK в «0» необходимо подать на вывод RESET положительный импульс длительностью не менее двух периодов тактового сигнала микроконтроллера.

   В моделях ATmega64x и ATmega128x напряжение НИЗКОГО уровня можно подавать на вывод REN вместо вывода RESET. Однако это возможно только в том случае, если программатор гарантирует удержание сигнала SCK в «0» при подаче питания. Кроме того, при использовании этого метода необходимо после программирования снять напряжение питания с микроконтроллера, чтобы он затем смог работать в нормальном режиме.

Таблица 4.16. **Команды режима программирования по последовательному каналу**

Название команды		Формат команды				Описание команды
		1-й байт	2-й байт	3-й байт	4-й байт	
Разрешение программирования		1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Разрешает программирование микроконтроллера, пока на выводе RESET присутствует напряжение НИЗКОГО уровня
Стирание кристалла		1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Очистка содержимого FLASH- и EEPROM-памяти. После посылки команды необходимо: 1. Выдержать паузу длительностью t_{WD_ERASE}. 2. Подать на вывод $\overline{\text{RESET}}$ положительный импульс. 3. Выждать не менее 20 мс. 4. Вновь послать команду «Разрешение программирования»
Чтение FLASH-памяти		0010 **H**000	**aaaa aaaa**	**bbbb bbbb**	**oooo oooo**	Чтение младшего (**H** = 0) или старшего (**H** = 1) байта памяти программ (**o**), расположенного по адресу **a**:**b**
Запись FLASH-памяти (семейство Tiny)		0100 **H**000	xxxx **aaaa**	**bbbb bbbb**	**iiii iiii**	Запись младшего (**H** = 0) или старшего (**H** = 1) байта (**i**) в память программ по адресу **a**:**b**
Загрузка страницы FLASH-памяти (семейство Mega)	8K	0100 **H**000	xxxx xxxx	xxx**b bbbb**	**iiii iiii**	Запись младшего (**H** = 0) или старшего (**H** = 1) байта (**i**) в буфер страницы памяти программ по адресу **b** (адрес слова в странице); младший байт должен загружаться первым
	16K			xx**bb bbbb**		
	32K			xx**bb bbbb**		
	64K			x**bbb bbbb**		
	128K			x**bbb bbbb**		

Продолжение таблицы 4.16

Название команды		Формат команды				Описание команды
		1-й байт	2-й байт	3-й байт	4-й байт	
Запись страницы FLASH-памяти (семейство Mega)	8K	0100 1100	xxxx **aaaa**	**bbb**x xxxx	xxxx xxxx	Запись страницы памяти программ по адресу **a**:**b**
	16K		xxx**a** **aaaa**	**bb**xx xxxx		
	32K		xx**aa** **aaaa**	**bb**xx xxxx		
	64K		x**aaa** **aaaa**	**b**xxx xxxx		
	128K		**aaaa** **aaaa**	**b**xxx xxxx		
Чтение EEPROM-памяти		1010 0000	xxxx **aaaa**	**bbbb bbbb**	**oooo oooo**	Чтение содержимого ячейки (**o**) EEPROM-памяти по адресу **a**:**b**
Запись EEPROM-памяти		1100 0000	xxxx **aaaa**	**bbbb bbbb**	**iiii iiii**	Запись значения (**i**) в ячейку EEPROM-памяти по адресу **a**:**b**
Загрузка страницы EEPROM-памяти (ATmega162x, страничный доступ)		1100 0001	0000 0000	0000 00**bb**	**iiii iiii**	Запись байта (**i**) в буфер страницы EEPROM-памяти по адресу **b** (адрес байта в странице)
Запись страницы EEPROM-памяти (ATmega162x, страничный доступ)		1100 0010	00xx xxx**a**	**bbbb bb**00	xxxx xxxx	Запись страницы EEPROM-памяти по адресу **a**:**b**
Чтение ячеек защиты		0101 1000	xxxx xxxx	xxxx xxxx	xx**oo** **oooo**	—
Запись ячеек защиты	ATtiny	1010 1100	1111 1ii1	xxxx xxxx	—	Запись ячеек защиты; для программирования ячейки соответствующий разряд (**i**) должен быть сброшен
	ATmega	1010 1100	111x xxxx	xxxx xxxx	—	
Чтение конфигурационных ячеек (для семейства Mega — младший байт)		0101 0000	0000 0000	xxxx xxxx	**oooo** **oooo**	Чтение конфигурационных ячеек; сброшенный разряд означает, что соответствующая конфигурационная ячейка запрограммирована
Чтение старшего конфигурационного байта (семейство Mega)		0101 0000	0000 1000	xxxx xxxx	**oooo** **oooo**	То же
Чтение дополнительного конфигурационного байта (семейство Mega)		0101 1000	0000 1000	xxxx xxxx	**oooo** **oooo**	То же

Продолжение таблицы 4.16

Название команды	Формат команды				Описание команды
	1-й байт	2-й байт	3-й байт	4-й байт	
Запись конфигурационных ячеек (для семейства Mega — младший байт)	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Запись конфигурационных ячеек. Для программирования ячейки соответствующий разряд (**i**) должен быть сброшен
Запись старшего конфигурационного байта (семейство Mega)	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	То же
Запись дополнительного конфигурационного байта (семейство Mega)	1010 1100	1010 0100	xxxx xxxx	**iiii iiii**	То же
Чтение идентификатора	0011 0000	xxxx xxxx	0000 00**bb**	**oooo oooo**	Чтение ячейки идентификатора (**o**) с адресом **b** (только в режимах защиты 1 и 2, см. **Табл. 4.2**)
Чтение калибровочного байта (ATmega)	0011 1000	xxxx xxxx	0000 00**bb**	**oooo oooo**	Чтение калибровочной константы (**o**) по адресу **b**

Примечание. Расшифровка условных обозначений, используемых в таблице:
a — разряды старшего байта адреса;
b — разряды младшего байта адреса;
H — «0» — младший байт, «1» — старший байт;
i — посылаемые в микроконтроллер данные;
o — считываемые из микроконтроллера данные;
x — состояние разряда безразлично.

2. Выждать не менее 20 мс.
3. Послать на вывод MOSI команду «Разрешение программирования».

Для контроля прохождения команды при посылке 3-го байта возвращается значение 2-го байта ($53). Если возвращаемое значение отлично от указанного, необходимо подать на вывод $\overline{\text{RESET}}$ положительный импульс и снова послать команду «Разрешение программирования». Причем независимо от возвращаемого значения необходимо передавать все 4 байта команды. Отсутствие возврата числа $53 после 32 попыток указывает на отсутствие связи между программатором и микросхемой либо на неисправность микросхемы.

После завершения программирования на вывод $\overline{\text{RESET}}$ можно подать напряжение ВЫСОКОГО уровня для перевода микроконтроллера в рабочий режим, либо выключить его. В последнем случае необходимо выполнить следующую последовательность действий:

1. Подать на вывод XTAL1 напряжение НИЗКОГО уровня, если тактирование микроконтроллера осуществляется от внешней схемы.
2. Подать на вывод RESET напряжение ВЫСОКОГО уровня.
3. Отключить напряжение питания от микроконтроллера.

23.3. Управлением процессом программирования FLASH-памяти

Программирование памяти программ микроконтроллеров семейства Tiny осуществляется побайтно путем посылки команд «Запись FLASH-памяти». В каждой команде передается адрес изменяемой ячейки и записываемое значение.

Следует помнить, что программирование ячейки можно выполнять только после завершения записи предыдущей ячейки. Для определения момента окончания записи существует два способа. Первый и наиболее универсальный способ — выдерживать между посылкой команд паузу, длительностью не меньше t_{WD_FLASH} (см. **Табл. 4.15**). Второй способ заключается в контролировании содержимого ячейки после посылки команды записи: до завершения записи ячейки при ее чтении возвращается значение «$FF», а после завершения — записанное значение.

Программирование памяти программ микроконтроллеров семейства Mega осуществляется постранично. Сначала содержимое страницы

побайтно заносится в буфер по командам «Загрузка страницы FLASH-памяти». В каждой команде передаются младшие разряды адреса изменяемой ячейки (положение ячейки внутри страницы) и записываемое значение. Содержимое каждой ячейки должно загружаться в следующей последовательности: сначала младший байт, потом старший.

Фактическое программирование страницы FLASH-памяти осуществляется после загрузки буфера страницы по команде «Запись страницы FLASH-памяти». В команде передаются старшие разряды адреса ячеек (номер страницы). Как и в семействе Tiny, дальнейшее программирование можно выполнять только после завершения записи страницы. Для определения момента окончания записи можно использовать любой из описанных выше способов. Заметим, что, поскольку запись всех ячеек страницы происходит одновременно, для определения момента окончания записи по второму способу можно использовать любую ячейку, находящуюся на странице.

23.4. Управление процессом программирования EEPROM-памяти

Программирование EEPROM-памяти данных микроконтроллеров семейства Tiny осуществляется побайтно путем посылки команд «Запись EEPROM-памяти». В каждой команде передается адрес изменяемой ячейки и записываемое значение. Как и для памяти программ, дальнейшее программирование можно выполнять только после завершения записи предыдущей ячейки. Для определения момента окончания записи можно либо выдерживать между посылкой команд паузу длительностью не меньше t_{WD_EEPROM} (см. **Табл. 4.15**), либо контролировать содержимое ячейки после посылки команды записи.

Во всех моделях семейства Mega программирование EEPROM-памяти осуществляется так же, как и в семействе Tiny, т. е. побайтно. Однако в моделях ATmega162x доступен альтернативный способ записи EEPROM-памяти — постраничный. Содержимое страницы побайтно заносится в буфер по командам «Загрузка страницы EEPROM-памяти», а затем осуществляется фактическое программирование страницы EEPROM-памяти по команде «Запись страницы FLASH-памяти». Для определения момента окончания записи можно использовать любой из описанных выше способов.

Глава 24. **Параллельное программирование**

24.1. Общие сведения

Режим параллельного программирования поддерживается моделями ATtiny28x семейства Tiny и всеми моделями микроконтроллеров семейства Mega. В этом режиме, как следует из его названия, от программатора к микроконтроллеру передаются одновременно все разряды кода команды или байта данных. Этот режим задействует большое число выводов микроконтроллера и, кроме того, требует использования дополнительного источника повышенного напряжения (12 В). Поэтому программирование в параллельном режиме осуществляется специализированными программаторами. Основное применение этого режима — «прошивка» микроконтроллеров перед установкой их на плату в условиях массового производства.

Схема включения микросхем в режиме параллельного программирования приведена на **Рис. 4.8.** Назначение сигналов, присутствующих на выводах микроконтроллера в этом режиме, приведено в **Табл. 4.17** и **Табл. 4.18**. Обратите внимание, что при последующем рассмотрении режима параллельного программирования выводы, указанные в **Табл. 4.17,** будут называться именами сигналов, присутствующих на этих выводах.

Временные диаграммы сигналов при программировании микроконтроллера в параллельном режиме представлены на **Рис. 4.9**, а значения параметров сигналов приведены в **Табл. 4.19**.

В общих чертах процесс программирования в этом режиме состоит из многократного выполнения следующих операций:

- загрузка команды;
- загрузка адреса;
- загрузка данных;
- выполнение команды.

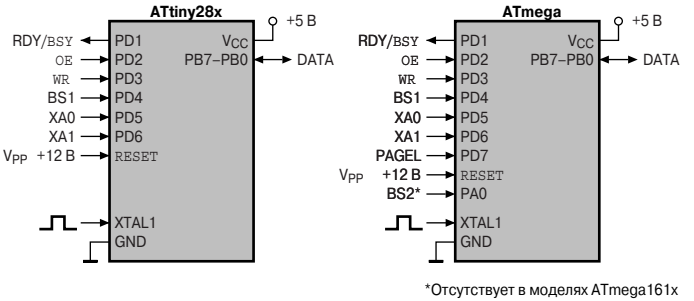


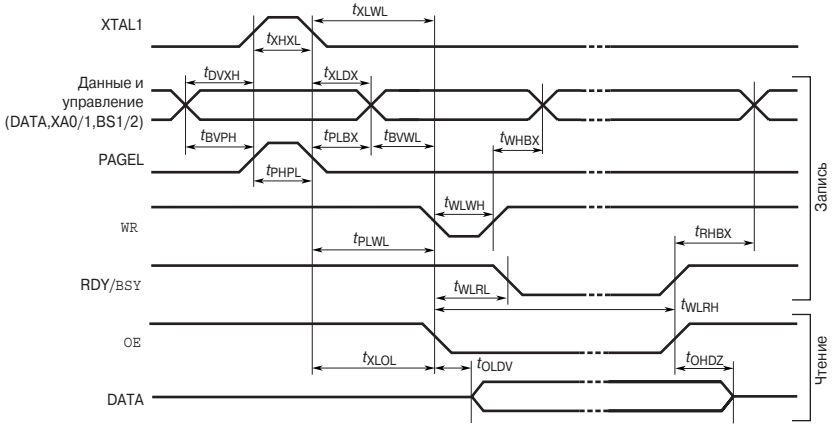
Рис. 4.8. Включение микроконтроллеров в режиме параллельного программирования

Таблица 4.17. **Обозначение и функции выводов, используемых при программировании в параллельном режиме**

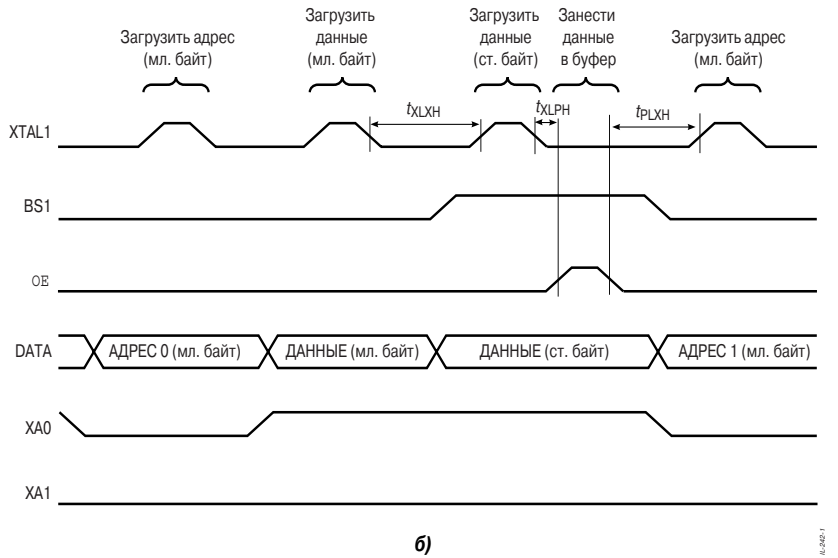
Сигнал	Вывод	Вход/Выход	Назначение
RDY/ $\overline{\text{BSY}}$	PD1	Выход	Состояние устройства: «0» — занято (выполняется предыдущая команда) «1» — готово к приему следующей команды
$\overline{\text{OE}}$	PD2	Вход	Управление режимом работы шины данных PB7...PB0: «0» — выход, «1» — вход
$\overline{\text{WR}}$	PD3	Вход	Сигнал записи (активный уровень — лог. 0)
BS1	PD4	Вход	Выбор байта («0» — младший байт, «1» — старший байт)
XA0	PD5	Вход	Определяют действие, выполняемое по положительному импульсу на выводе XTAL1 (**Табл. 4.18**)
XA1	PD6	Вход	
PAGEL*	PD7	Вход	Сигнал загрузки страницы памяти
BS2*	PA0	Вход	Выбор байта («0» — младший байт, «1» — старший байт)
DATA	PB7...PB0	Вход/Выход	Двунаправленная шина данных
* Эти сигналы используются только в микроконтроллерах семейства Mega.			

Таблица 4.18. **Функции сигналов XA0 и XA1**

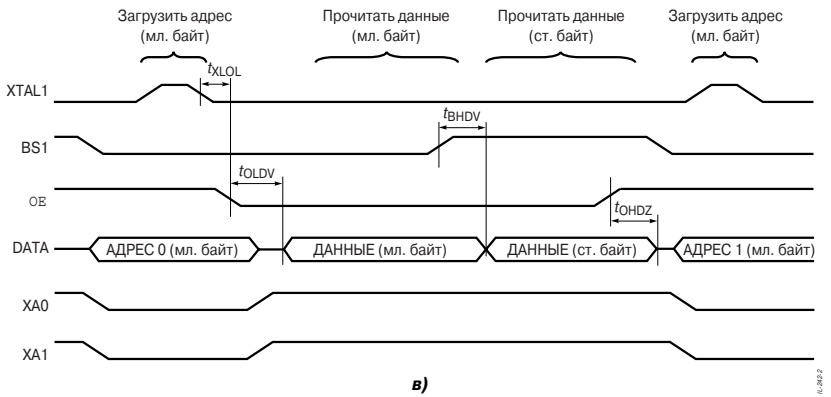
XA1	XA0	Действие, выполняемое по тактовому импульсу
0	0	Загрузка адреса ячейки памяти (младшего или старшего байта, в зависимости от уровня сигнала BS1)
0	1	Загрузка данных (младшего или старшего байта, в зависимости от уровня сигнала BS1)
1	0	Загрузка команды
1	1	Нет действия, режим ожидания


XTAL1
Данные и управление (DATA,XA0/1,BS1/2)
PAGEL
WR
RDY/BSY
OE
DATA
tXLWL
tXHXL
tDVXH
tXLDX
tBVPH
tPLBX
tBVWL
tWHBX
tPHPL
tWLWH
tPLWL
tRHBX
tWLRL
tWLRH
tXLOL
tOLDV
tOHDZ
Запись
Чтение
IL-842

а)


Загрузить адрес (мл. байт)
Загрузить данные (мл. байт)
Загрузить данные (ст. байт)
Занести данные в буфер
Загрузить адрес (мл. байт)
XTAL1
BS1
OE
DATA
XA0
XA1
tXLXH
tXLPH
tPLXH
АДРЕС 0 (мл. байт)
ДАННЫЕ (мл. байт)
ДАННЫЕ (ст. байт)
АДРЕС 1 (мл. байт)
IL-242-1

б)



в)

Рис. 4.9. Временные диаграммы сигналов при программировании в параллельном режиме:
а — общие требования; *б* — загрузка данных; *в* — чтение данных

Таблица 4.19. **Параметры сигналов при программировании в параллельном режиме**

Обозначение	Параметр	min	max
V_{PP}	Напряжение разрешения программирования [В]	11.5	12.5
I_{PP}	Ток, потребляемый от источника +12 В (VPP) [мкА]	—	250
t_{DVXH}	Задержка сигнала XTAL1 относительно момента установления сигналов управления и данных [нс]	67	—
t_{XLXH}	Время между импульсами сигнала XTAL1 [нс]	200	—
t_{XHXL}	Длительность импульсов сигнала XTAL1 [нс]	150	—
t_{XLDX}	Время удержания сигналов управления и данных относительно заднего фронта сигнала XTAL1 [нс]	67	—
t_{XLWL}	Задержка сигнала $\overline{WR}$ относительно заднего фронта сигнала XTAL1 [нс]	67	—
t_{XLPH}*	Задержка сигнала PAGEL относительно заднего фронта сигнала XTAL1 [нс]	0	—
t_{PLXH}*	Задержка сигнала XTAL1 относительно заднего фронта сигнала PAGEL [нс]	150	—
t_{BVPH}*	Задержка сигнала PAGEL относительно сигнала BS1 [нс]	67	—
t_{PHPL}*	Длительность импульса сигнала PAGEL [нс]	150	—
t_{PLBX}*	Время удержания сигнала BS1 относительно заднего фронта сигнала PAGEL [нс]	67	—
t_{WHBX}*	Время удержания сигналов BS1/BS2 относительно заднего фронта сигнала $\overline{WR}$ [нс]	67	—

Продолжение таблицы 4.19

Обозначение	Параметр	min	max
t_{PLWL}*	Задержка сигнала $\overline{WR}$ относительно заднего фронта сигнала PAGEL [нс]	67	—
t_{BVWL}	Задержка сигнала $\overline{WR}$ относительно момента установления сигнала BS1 [нс]	67	—
t_{RHBX}	Время удержания сигнала BS1 относительно заднего фронта сигнала RDY/ $\overline{BSY}$ [нс]	67	—
t_{WLWH}	Длительность импульса сигнала $\overline{WR}$ [нс]	150	—
t_{WLRL}	Задержка появления сигнала RDY/ $\overline{BSY}$ относительно переднего фронта сигнала $\overline{WR}$ [мкс]	0	2.5
t_{WLRH}	Задержка снятия сигнала RDY/ $\overline{BSY}$ относительно переднего фронта сигнала $\overline{WR}$ [мс]	0.5	15
t_{WLRH_CE}	Задержка снятия сигнала RDY/ $\overline{BSY}$ относительно переднего фронта сигнала $\overline{WR}$ для команды «Стирание кристалла» [мс]	7.5	30
t_{XLOL}	Задержка сигнала $\overline{OE}$ относительно заднего фронта сигнала XTAL1 [нс]	67	—
t_{BHDV}	Время установления сигналов данных относительно нарастающего фронта сигнала BS1 [нс]	0	250
t_{OLDV}	Время установления сигналов данных относительно переднего фронта сигнала $\overline{OE}$ [нс]	—	250
t_{OHDZ}	Задержка переключения шины данных в третье состояние относительно заднего фронта сигнала $\overline{OE}$ [нс]	—	250

* Только для микроконтроллеров семейства Mega.

Последовательность подачи сигналов на выводы микроконтроллера при выполнении различных базовых операций приведена в **Табл. 4.20**.

Таблица 4.20. **Базовые операции программирования в параллельном режиме**

№	Название операции	Действия
1	Загрузка команды	1. Установить выводы XA1, XA0 в состояние «10» (загрузка команды). 2. Подать на вывод BS1 напряжение лог. 0. 3. Выставить на шину DATA код команды (**Табл. 2.146**). 4. Подать на вывод XTAL1 положительный импульс
2	Загрузка адреса	1. Установить выводы XA1, XA0 в состояние «00» (загрузка адреса). 2. Подать на вывод BS1 напряжение лог. 0 (загрузка младшего байта) или лог. 1 (загрузка старшего байта). 3. Выставить на шину DATA младший байт адреса. 4. Подать на вывод XTAL1 положительный импульс

Продолжение таблицы 4.20

№	Название операции	Действия
3	Загрузка данных	1. Установить выводы XA1, XA0 в состояние «01» (загрузка данных). 2. В микроконтроллерах семейства Mega подать на вывод BS1 напряжение лог. 0 (загрузка младшего байта) или лог. 1 (загрузка старшего байта). 3. Выставить на шину DATA содержимое байта данных. 4. Подать на вывод XTAL1 положительный импульс
4	Запись данных в буфер (только ATmega)	1. Подать на вывод BS1 напряжение лог. 1. 2. Подать на вывод PAGEL положительный импульс
5	Запись ячейки памяти	1. Подать на вывод BS1 напряжение лог. 0 (запись младшего байта) или лог. 1 (запись старшего байта). 2. Подать на вывод $\overline{WR}$ отрицательный импульс; при этом на выводе RDY/ $\overline{BSY}$ появляется сигнал НИЗКОГО уровня. 3. Ждать появления на выводе RDY/ $\overline{BSY}$ сигнала ВЫСОКОГО уровня
	Запись страницы (только ATmega)	1. Подать на вывод BS1 напряжение лог. 0. 2. Подать на вывод $\overline{WR}$ отрицательный импульс; при этом на выводе RDY/ $\overline{BSY}$ появляется сигнал НИЗКОГО уровня. 3. Ждать появления на выводе RDY/ $\overline{BSY}$ сигнала ВЫСОКОГО уровня

В рассматриваемом режиме используется 9 команд, коды которых приведены в **Табл. 4.21**.

Таблица 4.21. **Команды программирования в параллельном режиме**

Код команды	Описание
1000 0000	«Стирание кристалла»
0100 0000	Запись конфигурационных ячеек
0010 0000	Запись ячеек защиты
0001 0000	Запись FLASH-памяти
0001 0001	Запись EEPROM-памяти
0000 1000	Чтение идентификатора
0000 0100	Чтение конфигурационных ячеек и ячеек защиты
0000 0010	Чтение FLASH-памяти*
0000 0011	Чтение EEPROM-памяти*
* Только в семействе Mega.	

24.2. Переключение в режим параллельного программирования

Первой операцией при программировании микроконтроллера является его перевод в режим программирования. Сразу следует сказать, что алгоритм, используемый для осуществления этой операции, зависит от модели микроконтроллера. Так, во всех микроконтроллерах семейства Tiny, а также в моделях ATmega161x, ATmega163x и ATmega323x семейства Mega для перевода микроконтроллера в режим программирования необходимо выполнить следующие действия:

1. Подать на микроконтроллер напряжение питания.
2. Подать на выводы RESET и BS1 напряжение НИЗКОГО уровня на время не менее 100 нс (500 нс для ATmega161x).
3. Подать напряжение 11.5...12.5 В на вывод RESET и удерживать напряжение НИЗКОГО уровня на выводе BS1 в течение как минимум 100 нс (500 нс для ATmega161x). Любая активность на выводе BS1 в течение этого времени приведет к тому, что микроконтроллер не перейдет в режим программирования.

В остальных микроконтроллерах семейства Mega для перевода микроконтроллера в режим программирования необходимо выполнить следующие действия:

1. Подать на микроконтроллер напряжение питания.
2. Подать на вывод RESET напряжение НИЗКОГО уровня и сформировать не меньше трех импульсов на выводе XTAL1.
3. Подать на выводы PAGEL, XA1, XA0, BS1 напряжение НИЗКОГО уровня на время не менее 100 нс.
4. Подать напряжение 11.5...12.5 В на вывод RESET и удерживать напряжение НИЗКОГО уровня на выводах PAGEL, XA1, XA0, BS1 в течение как минимум 100 нс. Любая активность на указанных выводах в течение этого времени приведет к тому, что микроконтроллер не перейдет в режим программирования.

Отдельно следует сказать о микроконтроллерах ATmega8x, т. к. в этих моделях вывод сброса может быть задействован под линию ввода/вывода (если конфигурационная ячейка RSTDISBL запрограммирована). В этом случае перед выполнением действий, описанных выше, необходимо сделать следующее:

1. Подать на выводы PAGEL, XA1, XA0, BS1 напряжение НИЗКОГО уровня.
2. Подать на микроконтроллер одновременно напряжение питания (VCC), а на вывод RESET — напряжение 11.5...12.5 В (VPP).
3. Выждать не менее 100 нс.

4. Перепрограммировать ячейку RSTDISBL (если установлена защита, то сначала необходимо выполнить стирание кристалла).
5. Выйти из режима программирования (отключить питание или подать на вывод RESET напряжение ВЫСОКОГО уровня).
6. Перевести микроконтроллер в режим программирования, как было описано ранее.

24.3. Стирание кристалла

Команда «Стирание кристалла» должна выполняться перед каждым перепрограммированием микроконтроллера. Данная команда полностью уничтожает содержимое FLASH- и EEPROM-памяти, а затем сбрасывает ячейки защиты (записывает в них «1»). Однако на состояние конфигурационных ячеек данная команда не влияет. Кроме того, в ряде моделей микроконтроллеров семейства Mega, можно предотвратить стирание EEPROM-памяти путем программирования конфигурационной ячейки EESAVE.

Для выполнения команды «Стирание кристалла» необходимо выполнить следующие действия:

1. Загрузить команду «Стирание кристалла» (код «1000 0000»).
2. Подать на вывод WR отрицательный импульс; при этом на выводе RDY/BSY появляется сигнал НИЗКОГО уровня.
3. Ждать появления на выводе RDY/BSY сигнала ВЫСОКОГО уровня.

24.4. Программирование FLASH-памяти

Запись FLASH-памяти

Сразу отметим, что алгоритм записи во FLASH-память зависит от семейства. Это связано с тем, что в семействе Tiny запись FLASH-памяти производится побайтно, а в семействе Mega — постранично.

Итак, в семействе Tiny запись FLASH-памяти производится в следующей последовательности (реализация каждого этапа приведена в **Табл. 4.20**):

1. Загрузить команду «Запись FLASH-памяти» (код «0001 0000»).
2. Загрузить старший байт адреса.
3. Загрузить младший байт адреса.
4. Загрузить младший байт данных.
5. Записать младший байт данных.

6. Загрузить старший байт данных.
7. Записать старший байт данных.

Временные диаграммы, иллюстрирующие процесс записи FLASH-памяти, приведены на **Рис. 4.10**.

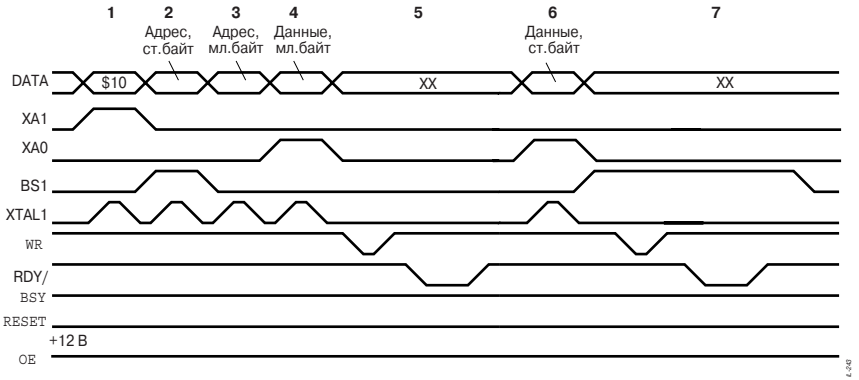


Рис. 4.10. Запись FLASH-памяти в микроконтроллерах семейства Tiny

Отметим, что загруженные код команды и адрес сохраняются в микроконтроллере до следующей загрузки новой команды или, соответственно, нового адреса. Из этого следует, что при программировании участка памяти команду необходимо загружать только один раз, старший байт адреса необходимо загружать только при переходе к новому 256-байтному блоку памяти. Кроме того, не требуется записывать значение «$FF», поскольку оно уже находится в ячейках памяти после выполнения команды «Стирание кристалла».

Следование этим рекомендациям позволит значительно ускорить процесс программирования.

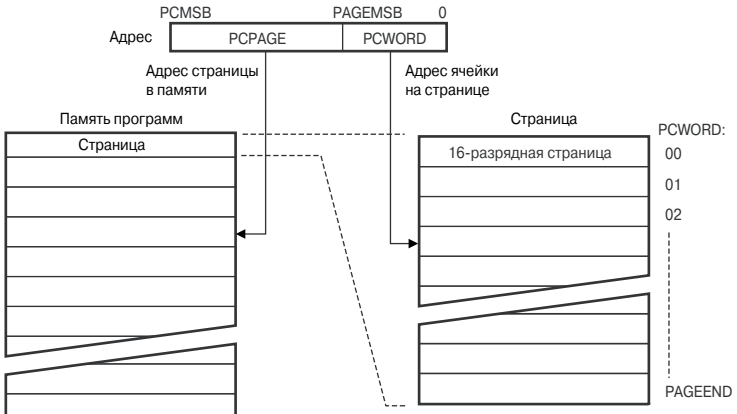
В семействе Mega запись FLASH-памяти производится в следующей последовательности (реализация каждого этапа процесса записи приведена в **Табл. 4. 21**):

1. Загрузить команду «Запись FLASH-памяти» (код «0001 0000»).
2. Загрузить младший байт адреса (положение ячейки внутри страницы).
3. Загрузить младший байт данных.
4. Загрузить старший байт данных.
5. Запомнить данные в буфере.
6. Повторить пп. 2...5 до полного заполнения буфера страницы.

7. Загрузить старший байт адреса (номер страницы).
8. Записать страницу.
9. Повторить пп. 2...8 для записи остальных страниц памяти программ.
10. Завершить программирование, загрузив команду «Нет операции» (код «0000 0000»).

Необходимо отметить, что для адресации ячейки памяти внутри страницы требуется меньше 8 разрядов (наибольший размер страницы составляет 128 слов). Оставшиеся старшие разряды младшего байта адреса используются для адресации страницы при выполнении команды «Запись страницы» (**Рис. 4.11**).

Временные диаграммы, иллюстрирующие процесс записи FLASH-памяти, приведены на **Рис. 4.12**.



Параметр	ATmega8x ATmega8515x	ATmega16x ATmega161x ATmega162x ATmega163x	ATmega32x ATmega323x	ATmega64x	ATmega128x
Размер памяти программ (в 16-разрядных словах)	4	8	16	32	64
PAGEEND	32	64	64	128	128
PCWORD	[4:0]	[0:6]	[0:5]	[0:5]	[0:6]
Кол-во страниц	128	128	256	256	512
PCPAGE	[11:5]	[12:6]	[13:6]	[14:7]	[15:7]

Рис. 4.11. Адресация FLASH-памяти микроконтроллеров семейства Mega при программировании

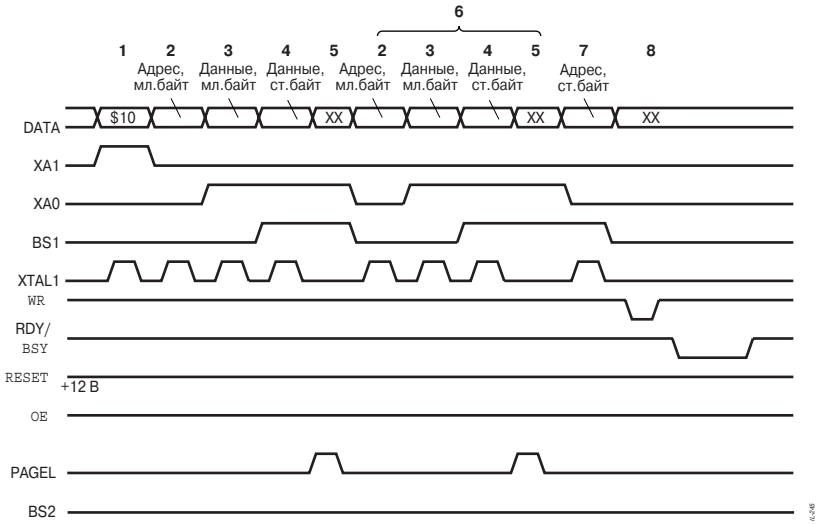


Рис. 4.12. Запись FLASH-памяти в микроконтроллерах семейства Mega

Чтение FLASH-памяти

Чтение FLASH-памяти в обоих семействах реализовано одинаково. Для осуществления чтения необходимо выполнить следующие действия (реализация каждого этапа приведена в **Табл. 4.20**):

1. Загрузить команду «Чтение FLASH-памяти» (код «0000 0010»).
2. Загрузить старший байт адреса.
3. Загрузить младший байт адреса.
4. Установить OE и BS1 в «0», после этого с шины данных DATA можно будет считать значение младшего байта содержимого ячейки памяти.
5. Установить BS1 в «1», после этого с шины данных DATA можно будет считать значение старшего байта содержимого ячейки памяти.
6. Установить OE в «1».

24.5. Программирование EEPROM-памяти

Запись EEPROM-памяти

В моделях ATmega161x, ATmega163x и ATmega323x запись EEPROM-памяти осуществляется побайтно. Для осуществления этой

операции необходимо выполнить следующие действия (реализация каждого этапа приведена в **Табл. 4.20**):

1. Загрузить команду «Запись EEPROM-памяти» (код «0001 0001»).
2. Загрузить старший байт адреса.
3. Загрузить младший байт адреса.
4. Загрузить младший байт данных.
5. Записать младший байт данных.

В остальных моделях семейства Mega EEPROM-память организована постранично, а для осуществления записи необходимо выполнить следующие действия:

1. Загрузить команду «Запись EEPROM-памяти» (код «0001 0001»).
2. Загрузить старший байт адреса.
3. Загрузить младший байт адреса.
4. Загрузить байт данных.
5. Запомнить данные в буфере.
6. Повторить пп. 3...5 до полного заполнения буфера.
7. Записать страницу.

Временные диаграммы, иллюстрирующие процесс записи EEPROM-памяти в этих моделях, приведены на **Рис. 4.13**.

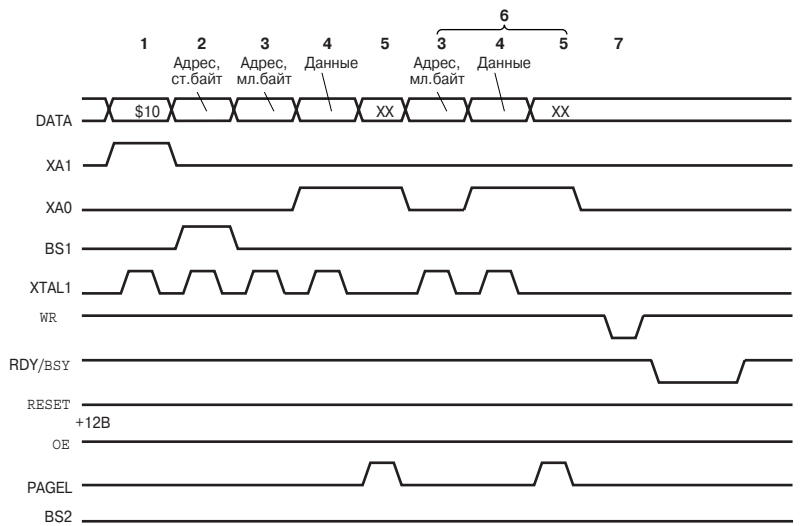


Рис. 4.13. Запись EEPROM-памяти со страничной организацией

Чтение EEPROM-памяти

Для чтения содержимого EEPROM-памяти необходимо выполнить следующие действия (реализация каждого этапа приведена в **Табл. 4.20**):

1. Загрузить команду «Чтение EEPROM-памяти» (код «0000 0011»).
2. Загрузить старший байт адреса.
3. Загрузить младший байт адреса.
4. Установить OE и BS1 в «0», после этого с шины данных DATA можно будет считать содержимое ячейки памяти.
5. Установить OE в «1».

24.6. Конфигурирование микроконтроллеров

24.6.1. Программирование конфигурационных ячеек

В микроконтроллере ATtiny28x семейства Tiny программирование конфигурационного байта выполняется в следующей последовательности (реализация каждого этапа приведена в **Табл. 4.20**):

1. Загрузить команду «Запись конфигурационных ячеек» (код «0100 0000»).
2. Загрузить байт данных. Если разряд сброшен в «0», выполняется программирование соответствующей ячейки, если установлен в «1» — ее сброс. Неиспользуемые разряды должны быть установлены в «1».
3. Записать младший байт данных.

В микроконтроллерах семейства Mega имеется уже по нескольку конфигурационных байтов, программирование которых выполняется в указанной ниже последовательности.

Младший конфигурационный байт

1. Загрузить команду «Запись конфигурационных ячеек» (код «0100 0000»).
2. Загрузить младший байт данных. Если разряд сброшен в «0», выполняется программирование соответствующей ячейки, если установлен в «1» — ее сброс.
3. Сбросить BS1 и BS2 в «0».
4. Подать на вывод WR отрицательный импульс и ждать появления на выводе RDY/ BSY сигнала ВЫСОКОГО уровня.

Старший конфигурационный байт

1. Загрузить команду «Запись конфигурационных ячеек» (код «0100 0000»).
2. Загрузить младший байт данных. Если разряд сброшен в «0», выполняется программирование соответствующей ячейки, если установлен в «1» — ее сброс.
3. Установить BS1 в «1», а BS2 в «0».
4. Подать на вывод WR отрицательный импульс и ждать появления на выводе RDY/ BSY сигнала ВЫСОКОГО уровня.
5. Сбросить BS1 в «0».

Дополнительный конфигурационный байт

1. Загрузить команду «Запись конфигурационных ячеек» (код «0100 0000»).
2. Загрузить младший байт данных. Если разряд сброшен в «0», выполняется программирование соответствующей ячейки, если установлен в «1» — ее сброс.
3. Установить BS1 в «0», а BS2 в «1».
4. Подать на вывод WR отрицательный импульс и ждать появления на выводе RDY/ BSY сигнала ВЫСОКОГО уровня.
5. Сбросить BS2 в «0».

24.6.2. Программирование ячеек защиты

Программирование ячеек защиты выполняется аналогично программированию конфигурационных ячеек (реализация каждого этапа приведена в **Табл. 4.23**):

1. Загрузить команду «Запись ячеек защиты» (код «0010 0000»).
2. Загрузить байт данных. Для программирования ячейки соответствующий разряд должен быть сброшен в «0». Неиспользуемые разряды должны быть всегда установлены в «1».
3. Записать младший байт данных.

24.6.3. Чтение конфигурационных ячеек и ячеек защиты

В микроконтроллерах ATtiny28x и ATmega161x чтение указанных ячеек выполняется в следующей последовательности (реализация каждого этапа приведена в **Табл. 4.20**):

1. Загрузить команду «Чтение конфигурационных ячеек и ячеек защиты» (код «0000 0100»).
2. Установить OE и BS1 в «0», после этого с шины данных DATA можно будет считать значение конфигурационного байта.

3. Установить BS1 в «1», после этого с шины данных DATA можно будет считать значение байта защиты.
4. Установить OE в «1».

В остальных микроконтроллерах семейства Mega эти же операции выполняются следующим образом (**Рис. 4.14**):

1. Загрузить команду «Чтение конфигурационных ячеек и ячеек защиты» (код «0000 0100»).
2. Установить OE, BS1 и BS2 в «0», после этого с шины данных DATA можно будет считать значение младшего конфигурационного байта.
3. Установить OE в «0», а BS1 и BS2 в «1». После этого с шины данных DATA можно будет считать значение старшего конфигурационного байта.
4. Установить OE в «0», BS1 в «0», а BS2 в «1». После этого с шины данных DATA можно будет считать значение дополнительного конфигурационного байта.
5. Установить OE в «0», BS1 в «1», а BS2 в «0». После этого с шины данных DATA можно будет считать значение байта защиты.
6. Установить OE в «1».

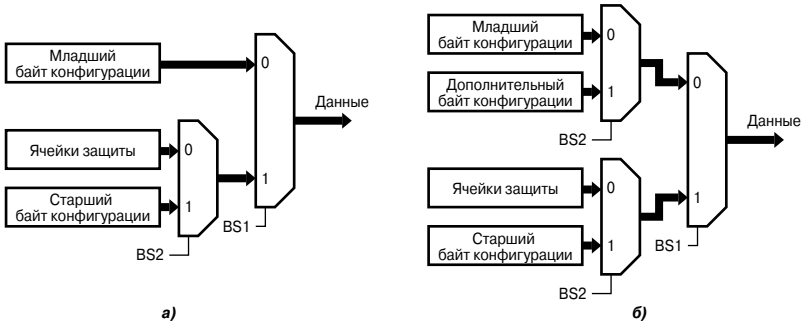


Рис. 4.14. Соответствие сигналов BS1 и BS2 выполняемой операции у модели с двумя байтами конфигурации (*а*) и модели с тремя байтами конфигурации (*б*)

24.6.4. Чтение ячеек идентификатора и калибровочной константы

Чтение ячеек идентификатора осуществляется в следующей последовательности (реализация каждого этапа приведена в **Табл. 4.20**):

1. Загрузить команду «Чтение ячеек идентификатора» (код «0000 1000»).
2. Загрузить младший байт адреса ($00...$02).

3. Установить OE и BS1 в «0», после этого с шины данных DATA можно будет считать содержимое выбранной ячейки идентификатора.
4. Установить OE в «1».

Чтение калибровочных констант осуществляется аналогичным образом и посредством той же команды:

1. Загрузить команду «Чтение ячеек идентификатора» (код «0000 1000»).
2. Загрузить младший байт адреса ($00...$03).
3. Установить OE в «0», а BS1 в «1», после этого с шины данных DATA можно будет считать значение выбранной калибровочной константы.
4. Установить OE в «1».

Глава 25. **Программирование по интерфейсу JTAG**

25.1. Общие сведения

Интерфейс JTAG был разработан группой ведущих специалистов по проблемам тестирования электронных компонентов (Joint Test Action Group). В дальнейшем он был зарегистрирован в качестве промышленного стандарта IEEE Std 1149.1-1990 (IEEE Standard Test Access Port and Boundary-Scan Architecture).

Встроенный в микроконтроллеры ATmega16x/162x/323x/64x/128x семейства Mega, интерфейс JTAG может быть использован для следующих целей:

- тестирования печатных плат;
- конфигурирования (программирования) кристалла;
- внутрисхемной отладки.

В данной книге будет рассмотрен только один из аспектов использования интерфейса JTAG, а именно программирование микроконтроллеров.

Доступ к модулю JTAG осуществляется через четыре вывода микроконтроллеров, составляющих так называемый «порт тестового доступа» (Test Access Port, TAP): TMS, TCK, TDI и TDO. Соответствие этих выводов контактам портов ввода/вывода микроконтроллера, а также их функции приведены в **Табл. 4.22**.

Таблица 4.22. **Выводы, используемые интерфейсом JTAG**

Название	ATmega 16x	ATmega 162x	ATmega 323x	ATmega 32x	ATmega 64x	ATmega 128x	Описание
TCK	PC2	PC4	PC2	PC2	PF4	PF4	Вход тактового сигнала
TMS	PC3	PC5	PC3	PC3	PF5	PF5	Вход выбора режима
TDO	PC4	PC6	PC4	PC4	PF6	PF6	Выход данных
TDI	PC5	PC7	PC5	PC5	PF7	PF7	Вход данных

Работой модуля JTAG управляет так называемый TAP-контроллер, представляющий собой конечный автомат с 16 состояниями. Диаграмма состояний TAP-контроллера приведена на **Рис. 4.15**. Переход между состояниями осуществляется по нарастающему фронту сигнала TCK в соответствии с сигналом, присутствующим на выводе TMS. После включения питания контроллер находится в состоянии «Test-Logic-Reset».

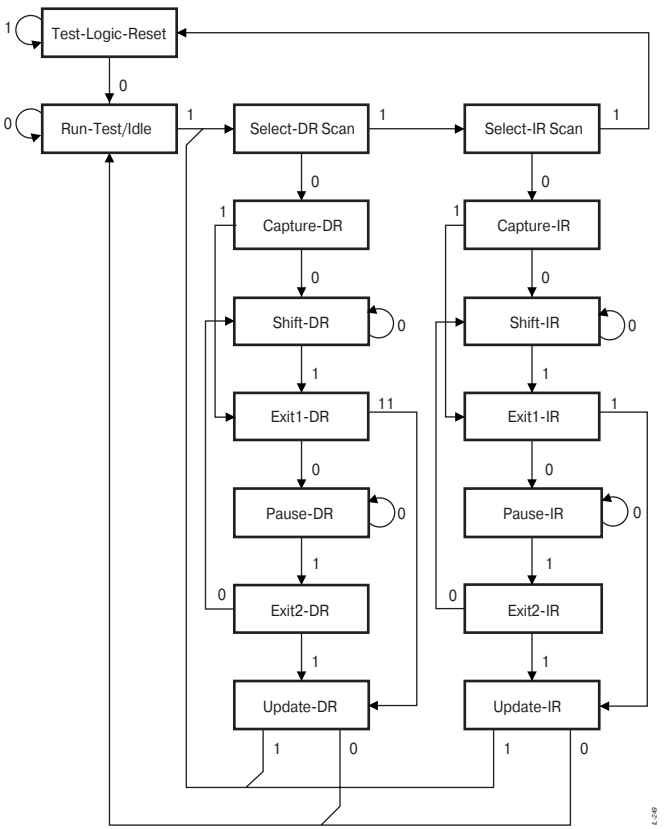


Рис. 4.15. Диаграмма состояний TAP-контроллера

Работа по интерфейсу JTAG осуществляется следующим образом (полагаем, что изначально TAP-контроллер находится в состоянии «Run-Test/Idle»):

- На вход TMS подается последовательность сигналов «1», «1», «0», «0», которые «защелкиваются» по нарастающему фронту сигнала TCK. В результате TAP-контроллер переходит в состояние «Shift-IR», в котором к входу TDI подключается сдвиговый регистр команд. После этого на вход TDI подается начиная с младшего разряда 4-разрядный код команды. Занесение разрядов кода в регистр команд также осуществляется по нарастающему фронту сигнала TCK. При загрузке 3-х младших разрядов на выводе TMS должен удерживаться сигнал лог. 0, чтобы TAP-контроллер оставался в состоянии «Shift-IR». Старший разряд кода команды заносится в регистр при выходе контроллера из этого состояния, осуществляемого подачей на вход TMS лог. 1. При этом контроллер переходит в состояние «Exit-IR». Во время загрузки команды на вывод TDO выдается значение $01 (начиная с младшего разряда). Загруженная команда определяет, какой из регистров данных JTAG будет далее подключен между выводами TDI и TDO, а также управляет элементами микроконтроллера, с которыми связан этот регистр данных.
- На вход TMS подается последовательность сигналов «1», «0», в результате чего микроконтроллер возвращается в состояние «Run-Test/Idle». При прохождении состояния «Update-IR» содержимое сдвигового регистра фиксируется на его параллельном выходе.
- На вход TMS подается последовательность сигналов «1», «0», «0», которые «защелкиваются» по нарастающему фронту сигнала TCK. В результате TAP-контроллер переходит в состояние «Shift-DR». В этом состоянии в регистр данных, определяемый загруженной ранее командой, заносятся начиная с младшего разряда необходимые данные. Как и в случае команд, загрузка осуществляется с вывода TDI по нарастающему фронту сигнала TCK. При загрузке всех разрядов регистра, кроме самого старшего, на выводе TMS должен удерживаться сигнал лог. 0, чтобы TAP-контроллер оставался в состоянии «Shift-DR». Старший разряд данных заносится в регистр при выходе контроллера из этого состояния, осуществляемого подачей на вход TMS лог. 1. При этом контроллер переходит в состояние «Exit-DR». Одновременно с загрузкой данных на вывод TDO выдается начиная с младшего разряда содержимое регистра данных, «захваченного» в состоянии «Capture-DR».
- На вход TMS подается последовательность сигналов «1», «0», в результате чего микроконтроллер вновь возвращается в состояние «Run-Test/Idle». Если выбранный регистр имеет параллель-

ный выход, то его содержимое фиксируется на нем при прохождении состояния «Update-DR».

Остается только добавить, что независимо от начального состояния TAP-контроллера он всегда возвращается в состояние «Test-Logic-Reset» после удержания на выводе TMS сигнала лог. 1 в течение 5 периодов сигнала TCK.

25.2. Использование интерфейса JTAG для программирования кристалла. Команды JTAG

Разрешение/запрещение интерфейса JTAG (причем не только для программирования) осуществляется конфигурационной ячейкой JTAGEN. Если она не запрограммирована («1»), выводы TAP работают как обычные контакты портов ввода/вывода, а TAP-контроллер находится в состоянии сброса. Для включения интерфейса ячейка JTAGEN должна быть запрограммирована (состояние по умолчанию). Кроме того, должен быть сброшен разряд JTD регистра управления MCUCSR (**Рис. 4.16**). Сброс этого разряда может быть осуществлен как программно, записью в этот разряд лог. 0, так и аппаратно, подачей на вывод $\overline{\text{RESET}}$ напряжения НИЗКОГО уровня. Причем для программного изменения состояния этого разряда новое значение в него необходимо записать дважды в течение 4-х машинных циклов.

	7	6	5	4	3	2	1	0
	JTD	X	X	JTRF	WDRF	BORF	EXTRF	PORF
Чтение(R)/Запись(W)	R/W	X	X	R/W	R/W	R/W	R/W	R/W
Начальное значение	0	0	0	0	X	X	X	X

Рис. 4.16. Регистр MCUCSR применительно к интерфейсу JTAG

Описанный механизм позволяет использовать выводы TAP как в качестве контактов портов ввода/вывода при нормальном функционировании микроконтроллера, так и в качестве выводов собственно порта JTAG при программировании кристалла. Разумеется, этот механизм не применим в том случае, если порт JTAG используется для отладки или тестирования.

Из 16 команд, поддерживаемых интерфейсом, при программировании используются только пять. Описания этих команд приведены далее.

25.2.1. AVR_RESET (код команды $0C)

Эта команда предназначена для перевода микроконтроллера в состояние сброса и, соответственно, вывода его из этого состояния. В качестве регистра данных выбирается 1-разрядный регистр сброса (Reset Register). Запись «1» в этот регистр эквивалентна подаче на вывод RESET микроконтроллера напряжения НИЗКОГО уровня. В состоянии сброса микроконтроллер будет находиться до тех пор, пока в регистр сброса не будет записан «0».

Активные состояния:

Shift-DR — осуществляется загрузка регистра сброса.

25.2.2. PROG_ENABLE (код команды $04)

Эта команда предназначена для разрешения программирования кристалла через порт JTAG. В качестве регистра данных выбирается 16-разрядный регистр разрешения программирования (Programming Enable Register).

При записи в этот регистр числа $A370 (сигнатура разрешения программирования) разрешается программирование микроконтроллера по интерфейсу JTAG. При выходе из режима программирования этот регистр должен сбрасываться.

Активные состояния:

Shift-DR — осуществляется загрузка регистра разрешения прерывания;

Update-DR — осуществляется сравнение содержимого регистра с числом $A370 и в случае совпадения перевод микроконтроллера в режим программирования.

25.2.3. PROG_COMMANDS (код команды $05)

Эта команда предназначена для загрузки команд программирования и выдачи результатов их выполнения (если они есть). В качестве регистра данных выбирается 15-разрядный регистр команд (Programming Command Register).

Активные состояния:

Capture-DR — результат выполнения предыдущей команды загружается в регистр;

Shift-DR — по нарастающему фронту сигнала TCK осуществляется выдача результата с одновременной загрузкой новой команды;

Update-DR — загруженная команда подается на блок памяти микроконтроллера;
Run-Test/Idle — в ряде случаев генерируется один тактовый импульс, необходимый для выполнения команды.

25.2.4. PROG_PAGELOAD (код команды $06)

Эта команда предназначена для непосредственной загрузки страницы памяти программ через порт JTAG. В качестве регистра данных выбирается регистр загрузки виртуальной страницы (Virtual Flash Page Load Register), размер которого равен размеру одной страницы памяти программ. Собственно сдвиговый регистр является 8-разрядным, а побайтная пересылка данных в буфер осуществляется автоматически.

Активные состояния:

Shift-DR — осуществляется поразрядная загрузка данных и побайтная их пересылка в буфер страницы FLASH-памяти.

25.2.5. PROG_PAGEREAD (код команды $07)

Эта команда предназначена для считывания содержимого страницы памяти программ через порт JTAG. В качестве регистра данных выбирается регистр чтения виртуальной страницы (Virtual Flash Page Read Register), размер которого на 1 байт больше размера страницы памяти программ. Побайтная пересылка содержимого страницы в сдвиговый регистр осуществляется автоматически.

Активные состояния:

Shift-DR — осуществляется побайтная пересылка содержимого страницы FLASH-памяти в сдвиговый регистр и поразрядная его выдача на вывод TDO. Первые 8 тактов используются для первоначальной загрузки сдвигового регистра, поэтому выдаваемые в это время разряды должны быть проигнорированы.

5.2.6. Алгоритм программирования

В этом параграфе описываются действия, которые необходимо выполнить для программирования микроконтроллеров через порт JTAG. Формат всех команд, используемых при программировании (не путать с командами JTAG), приведен в **Табл. 4.23**.

Таблица 4.23. **Команды программирования по интерфейсу JTAG**

Команда		TDI	TDO	Прим.
1	a. Стирание кристалла	0100011_10000000 0110001_10000000 0110011_10000000 0110011_10000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	—
	b. Запрос состояния операции	0110011_10000000	xxxxx**o**x_xxxxxxxx	2
2	a. Вход в режим записи FLASH-памяти	0100011_00010000	xxxxxxx_xxxxxxxx	—
	b. Загрузка старшего байта адреса	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	6
	c. Загрузка младшего байта адреса	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	—
	d. Загрузка младшего байта данных	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	—
	e. Загрузка старшего байта данных	0010111_**iiiiiiii**	xxxxxxx_xxxxxxxx	—
	f. Фиксация данных	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	—
	g. Запись страницы FLASH-памяти	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	—
	h. Запрос состояния операции	0110111_00000000	xxxxx**o**x_xxxxxxxx	2
3	a. Вход в режим чтения FLASH-памяти	0100011_00000010	xxxxxxx_xxxxxxxx	—
	b. Загрузка старшего байта адреса	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	6
	c. Загрузка младшего байта адреса	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	—
	d. Чтение младшего и старшего байтов данных	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo** xxxxxxx_**oooooooo**	 8 9

Продолжение таблицы 4.23

	Команда	TDI	TDO	Прим.
4	a. Вход в режим записи EEPROM-памяти	0100011_00010001	xxxxxxx_xxxxxxxx	—
	b. Загрузка старшего байта адреса	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	6
	c. Загрузка младшего байта адреса	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	—
	d. Загрузка байта данных	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	—
	e. Фиксация данных	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	—
	f. Запись страницы EEPROM-памяти	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	—
	g. Запрос состояния операции	0110011_00000000	xxxxx**o**x_xxxxxxxx	2
5	a. Вход в режим чтения EEPROM-памяти	0100011_00000011	xxxxxxx_xxxxxxxx	—
	b. Загрузка старшего байта адреса	0000111_**aaaaaaaa**	xxxxxxx_xxxxxxxx	6
	c. Загрузка младшего байта адреса	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	—
	d. Чтение байта данных	0110011_**bbbbbbbb** 0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	—

Продолжение таблицы 4.23

	Команда	TDI	TDO	Прим.
6	a. Вход в режим записи конфигурационных ячеек	0100011_01000000	xxxxxxx_xxxxxxxx	—
	b. Загрузка байта данных	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	3; 5
	c. Запись дополнительного байта конфигурации	0111011_00000000 0111001_00000000 0111011_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	—
	d. Запрос состояния операции	0110111_00000000	xxxxx**o**x_xxxxxxxx	2
	e. Загрузка байта данных	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	3; 5
	f. Запись старшего байта конфигурации	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	—
	g. Запрос состояния операции	0110111_00000000	xxxxx**o**x_xxxxxxxx	2
	h. Загрузка байта данных	0010011_**iiiiiiii**	xxxxxxx_xxxxxxxx	3; 5
	i. Запись младшего байта конфигурации	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	—
	j. Запрос состояния операции	0110011_00000000	xxxxx**o**x_xxxxxxxx	2
7	a. Вход в режим записи ячеек защиты	0100011_00100000	xxxxxxx_xxxxxxxx	—
	b. Загрузка байта данных	0010011_11**iiiiii**	xxxxxxx_xxxxxxxx	3; 6
	c. Запись байта защиты	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	—
	d. Запрос состояния операции	0110011_00000000	xxxxx**o**x_xxxxxxxx	2

Продолжение таблицы 4.23

	Команда	TDI	TDO	Прим.
8	a. Вход в режим чтения конфигурационных ячеек и ячеек защиты	0100011_00000100	xxxxxxx_xxxxxxxx	—
	b. Чтение дополнительного байта конфигурации	0111010_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	5
	c. Чтение старшего байта конфигурации	0111110_00000000 0111111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	5
	d. Чтение младшего байта конфигурации	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	5
	e. Чтение байта защиты	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xx**oooooo**	4; 6
	f. Чтение конфигурационных ячеек и ячеек защиты	0111010_00000000 0111110_00000000 0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo** xxxxxxx_**oooooooo** xxxxxxx_**oooooooo** xxxxxxx_xx**oooooo**	4 7 9 8 10
9	a. Вход в режим чтения ячеек идентификатора	0100011_00001000	xxxxxxx_xxxxxxxx	—
	b. Загрузка байта адреса	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	—
	c. Чтение байта идентификатора	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	—
10	a. Вход в режим чтения калибровочной ячейки	0100011_00001000	xxxxxxx_xxxxxxxx	—
	b. Загрузка байта адреса	0000011_**bbbbbbbb**	xxxxxxx_xxxxxxxx	—
	c. Чтение калибровочной константы	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_**oooooooo**	—
11	a. Загрузка команды «Нет операции»	0100011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	—

Примечания: 1. Условные обозначения, используемые в таблице:
a — разряды старшего байта адреса;
b — разряды младшего байта адреса;
i — посылаемые в микроконтроллер данные;
o — считываемые из микроконтроллера данные;
x — состояние разряда безразлично.
2. Повторять до тех пор, пока **o** = «1».
3. Для программирования ячейки соответствующий ей разряд должен быть сброшен, для стирания — установлен.
4. «0» — ячейка запрограммирована, «1» — не запрограммирована.
5. Соответствие разрядов конфигурационным ячейкам — см. **Табл. 4.6**.
6. Соответствие разрядов ячейкам защиты — см. **Рис. 4.1**.
7. Дополнительный байт.
8. Младший байт.
9. Старший байт.
10. Байт защиты.

Ниже приведен порядок выполнения 12 операций.

Операция «*Вход в режим программирования*»:
1. Загрузить команду AVR_RESET и занести «1» в регистр сброса.
2. Загрузить команду PROG_ENABLE и занести в регистр разрешения программирования значение 1010_0011_0111_0000 ($A370).

Операция «*Выход из режима программирования*»:
1. Загрузить команду PROG_COMMANDS.
2. Запретить выполнение всех команд программирования, запустив команду «Нет операции» (11a).
3. Загрузить команду PROG_ENABLE и занести в регистр разрешения программирования значение 0000_0000_0000_0000.
4. Загрузить команду AVR_RESET и занести «0» в регистр сброса.

Операция «*Стирание кристалла*»:
1. Загрузить команду PROG_COMMANDS.
2. Начать стирание кристалла, запустив команду 1a.
3. Дождаться окончания выполнения этой операции, контролируя значение, возвращаемое командой 1b. В качестве альтернативы можно просто выждать время t_{WLRH_CE} (см. **Табл. 4.19**).

Операция «*Программирование FLASH-памяти*»:
1. Загрузить команду PROG_COMMANDS.
2. Войти в режим программирования FLASH-памяти (команда 2a).
3. Загрузить старший байт адреса (команда 2b).
4. Загрузить младший байт адреса (команда 2c).
5. Загрузить данные (команды 2d, 2e и 2f).
6. Повторить п.п. 4 и 5 для каждой ячейки страницы памяти программ.
7. Записать страницу (команда 2g).
8. Дождаться окончания выполнения этой операции, контролируя значение, возвращаемое командой 2h. В качестве альтернативы можно просто выждать время t_{WLRH} (см. **Табл. 4.19**).
9. Повторить пп. 3...8 для программирования остальных страниц памяти программ.

Существует также другой, более эффективный способ загрузки данных в микроконтроллер, основанный на использовании команды PROG_PAGELOAD:
1. Загрузить команду PROG_COMMANDS.
2. Войти в режим программирования FLASH-памяти (команда 2a).

3. Загрузить адрес страницы, используя команды 2b и 2c.
4. Загрузить команду PROG_PAGELOAD.
5. Загрузить содержимое страницы, начиная с МЗР первой ячейки на странице и заканчивая СЗР последней ячейки.
6. Загрузить команду PROG_COMMANDS.
7. Записать страницу (команда 2g).
8. Дождаться окончания выполнения этой операции, контролируя значение, возвращаемое командой 2h. В качестве альтернативы можно просто выждать время t_{WLRH} (см. **Табл. 4.19**).
9. Повторить пп. 3...8 для программирования остальных страниц памяти программ.

Операция «*Чтение FLASH-памяти*»:
1. Загрузить команду PROG_COMMANDS.
2. Войти в режим чтения FLASH-памяти (команда 3a).
3. Загрузить адрес ячейки памяти (команды 3b и 3c).
4. Считать содержимое ячейки (команда 3d).
5. Повторить пп. 3 и 4 для чтения остальных ячеек памяти программ.

Для выполнения рассматриваемой операции можно также воспользоваться командой PROG_PAGEREAD:
1. Загрузить команду PROG_COMMANDS.
2. Войти в режим чтения FLASH-памяти (команда 3a).
3. Загрузить адрес страницы памяти программ (команды 3b и 3c).
4. Разряды PCWORD (см. **Рис. 4.11**) должны быть равны «0».
5. Загрузить команду PROG_PAGEREAD.
6. Считать содержимое страницы начиная с МЗР первой ячейки на странице и заканчивая СЗР последней ячейки. Напоминаем, что первые 8 разрядов должны быть проигнорированы.
7. Загрузить команду PROG_COMMANDS.
8. Повторить пп. 3...6 для чтения остальных страниц памяти программ.

Операция «*Программирование EEPROM-памяти*»:
1. Загрузить команду PROG_COMMANDS.
2. Войти в режим процесса программирования EEPROM-памяти (команда 4a).
3. Загрузить старший байт адреса (команда 4b).
4. Загрузить младший байт адреса (команда 4c).
5. Загрузить данные (команды 4d и 4e).

6. Повторить пп. 4 и 5 для всех ячеек страницы.
7. Записать страницу (команда 4f).
8. Дождаться окончания выполнения этой операции, контролируя значение, возвращаемое командой 4g. В качестве альтернативы можно просто выждать время t_{WLRH} (см. **Табл. 4.19**).
9. Повторить пп. 3...8 для программирования остальных страниц EEPROM-памяти.

Операция «*Чтение EEPROM-памяти*»:
1. Загрузить команду PROG_COMMANDS.
2. Войти в режим чтения EEPROM-памяти (команда 5a).
3. Загрузить адрес ячейки памяти (команда 5b и 5c).
4. Считать содержимое ячейки (команда 5d).
5. Повторить пп. 3 и 4 для чтения остальных ячеек EEPROM-памяти.

Операция «*Программирование конфигурационных ячеек*»:
1. Загрузить команду PROG_COMMANDS.
2. Войти в режим программирования конфигурационных ячеек (команда 6a).
3. Загрузить байт данных (команда 6b).
4. Записать дополнительный байт конфигурации (команда 6c).
5. Дождаться окончания выполнения этой операции, контролируя значение, возвращаемое командой 6d. В качестве альтернативы можно просто выждать время t_{WLRH} (см. **Табл. 4.19**).
6. Загрузить байт данных (команда 6e).
7. Записать старший байт конфигурации (команда 6f).
8. Дождаться окончания выполнения этой операции, контролируя значение, возвращаемое командой 6g. В качестве альтернативы можно просто выждать время t_{WLRH} (см. **Табл. 4.19**).
9. Загрузить байт данных (команда 6h).
10. Записать младший байт конфигурации (команда 6i).

Операция «*Программирование ячеек защиты*»:
1. Загрузить команду PROG_COMMANDS.
2. Войти в режим программирования ячеек защиты(команда 7a).
3. Загрузить байт данных (команда 7b).
4. Записать байт защиты (команда 7c).
5. Дождаться окончания выполнения этой операции, контролируя значение, возвращаемое командой 7d. В качестве альтернативы можно просто выждать время t_{WLRH} (см. **Табл. 4.19**).

Операция «*Чтение конфигурационных ячеек и ячеек защиты*»:
1. Загрузить команду PROG_COMMANDS.
2. Войти в режим чтения конфигурационных ячеек и ячеек защиты (команда 8a).
3. Войти в режим чтения всех ячеек (команда 8f).
4. Войти в режим чтения только дополнительного байта конфигурации (команда 8b).
5. Войти в режим чтения только старшего байта конфигурации (команда 8c).
6. Войти в режим чтения только младшего байта конфигурации (команда 8d).
7. Войти в режим чтения только байта защиты (команда 8e).

Операция «*Чтение ячеек идентификатора*»:
1. Загрузить команду PROG_COMMANDS.
2. Войти в режим чтения ячеек идентификатора (команда 9a).
3. Загрузить адрес $00, используя команду 9b.
4. Прочитать первый байт идентификатора (команда 9c).
5. Повторить пп. 3 и 4 для чтения второго (адрес $01) и третьего (адрес $02) байта идентификатора.

Операция «*Чтение калибровочных ячеек*»:
1. Загрузить команду PROG_COMMANDS.
2. Войти в режим чтения калибровочных ячеек (команда 10a).
3. Загрузить адрес ячейки (команда 10b).
4. Считать значение калибровочной константы (команда 10c).

Глава 26. Самопрограммирование микроконтроллеров семейства Mega

26.1. Общие сведения

Все микроконтроллеры семейства Mega имеют возможность самопрограммирования, т. е. самостоятельного изменения содержимого своей памяти программ. Эта особенность позволяет создавать на их основе очень гибкие системы, алгоритм работы которых будет меняться самим микроконтроллером в зависимости от каких-либо внутренних условий или внешних событий.

Для поддержки самопрограммирования вся область памяти программ логически разделена на две секции — секцию прикладной программы (Application Section) и секцию загрузчика (Boot Loader Section). Изменение памяти программ осуществляется программой-загрузчиком, расположенной в одноименной секции. Для загрузки нового содержимого памяти программ, а также для выгрузки старого содержимого программа-загрузчик может использовать любой интерфейс передачи данных (USART/UART, SPI, TWI), имеющийся в составе конкретного микроконтроллера. Сразу отметим, что загрузчик может изменять содержимое обеих секций. Это позволяет ему модифицировать собственный код и даже удалять себя из памяти, если надобность в нем отпадет. Уровень доступа (чтение/запись) к каждой из секций задается пользователем с помощью ячеек защиты BLB02:BLB01 и BLB12:BLB11, описанных в 1-й главе этой части (см. **Табл. 4.2**).

Переход к программе-загрузчику может осуществляться различным образом. В частности, она может быть вызвана из основной программы командами CALL/JMP. Отметим, что в моделях ATmega161x этот способ является единственным. Другим способом является перемещение вектора сброса в начало секции загрузчика. В этом случае запуск программы-загрузчика будет осуществляться автоматически после каждого сброса микроконтроллера. Положение вектора сброса определя-

ется состоянием конфигурационной ячейки BOOTRST. Если в ней содержится «1», вектор сброса располагается в начале памяти программ по адресу $0000. При запрограммированнной ячейке, когда в ней содержится «0», вектор сброса располагается в начале секции загрузчика (адреса см. **Табл. 4.24**).

Размер секции загрузчика и размер секции прикладной программы практически во всех микроконтроллерах задается с помощью двух конфигурационных ячеек BOOTSZ1:BOOTSZ0. Исключение составляют лишь модели ATmega161x, в которых размеры секций являются фиксированными. Возможные конфигурации памяти программ всех микроконтроллеров семейства приведены в **Табл. 4.24**.

Таблица 4.24. **Конфигурация памяти программ**

Модель	BOOTSZ1	BOOTSZ0	Размер загрузчика [слов]	Страниц	Секция прикладной программы	Секция загрузчика
ATmega8x ATmega8515x	1	1	128	4	$000...$F7F	$F80...$FFF
	1	0	256	8	$000...$EFF	$F00...$FFF
	0	1	512	16	$000...$DFF	$E00...$FFF
	0	0	1024	32	$000...$BFF	$C00...$FFF
ATmega16x ATmega162x ATmega163x	1	1	128	2	$0000...$1F7F	$1F80...$1FFF
	1	0	256	4	$0000...$1EFF	$1F00...$1FFF
	0	1	512	8	$0000...$1DFF	$1E00...$1FFF
	0	0	1024	16	$0000...$1BFF	$1C00...$1FFF
ATmega161x	—	—	512	8	$0000...$1DFF	$1E00...$1FFF
ATmega323x	1	1	256	4	$0000...$3EFF	$3F00...$3FFF
	1	0	512	8	$0000...$3DFF	$3E00...$3FFF
	0	1	1024	16	$0000...$3BFF	$3C00...$3FFF
	0	0	2048	32	$0000...$37FF	$3800...$3FFF
ATmega64x	1	1	512	4	$0000...$7DFF	$7E00...$7FFF
	1	0	1024	8	$0000...$7BFF	$7C00...$7FFF
	0	1	2048	16	$0000...$77FF	$7800...$7FFF
	0	0	4096	32	$0000...$6FFF	$7000...$7FFF
ATmega64x	1	1	512	4	$0000...$FDFF	$FE00...$FFFF
	1	0	1024	8	$0000...$FBFF	$FC00...$FFFF
	0	1	2048	16	$0000...$F7FF	$F800...$FFFF
	0	0	4096	32	$0000...$FFFF	$F000...$FFFF

26.2. Области RWW и NRWW

В микроконтроллерах ATmega8x, ATmega8515x, ATmega16x, ATmega162x и ATmega64x/128x имеется также и другое разделение памяти программ. В этих моделях вся память программ разбита на две области фиксированного размера, называемых «чтение при записи» (Read-While-Write, RWW) и «нет чтения при записи» (No Read-While-Write, NRWW). Размеры этих областей для всех микроконтроллеров семейства приведены в **Табл. 4.25**.

Таблица 4.25. **Области RWW и NRWW**

Секция	ATmega8x ATmega8515x		ATmega16x ATmega162x		ATmega64x		ATmega128x	
	*	Адреса	*	Адреса	*	Адреса	*	Адреса
RWW	96	$000...$BFF	112	$0000...$1BFF	224	$0000...$6FFF	480	$0000...$EFFF
NRWW	32	$C00...$FFF	16	$1C00...$1FFF	32	$7000...$7FFF	32	$F000...$FFFF
* Количество страниц								

Отличие между этими областями заключается в различном поведении центрального процессора при изменении расположенных в них данных:

- во время выполнения операции стирания или записи страницы памяти программ, расположенной в области RWW, процессор может осуществлять чтение только из области NRWW;
- во время выполнения операции стирания или записи страницы памяти программ, расположенной в области NRWW, процессор останавливается до окончания этой операции.

Таким образом, во время изменения содержимого страницы памяти программ, расположенной в области RWW, чтение этой области запрещено. Попытка обратиться во время программирования к коду, находящемуся в области RWW (в результате выполнения команд CALL/JMP/LPM или в результате прерывания), может привести к непредсказуемым последствиям. Во избежание этого следует либо запретить прерывания, либо перенести таблицу векторов прерываний в секцию загрузчика, которая всегда находится в области NRWW.

Для определения того, разрешено чтение из области RWW или нет, предназначен флаг RWWSB регистра SPMCR (см. ниже). Установленный в «1» флаг означает, что область RWW заблокирована для чтения. По окончании операции программирования флаг RWWSB должен быть сброшен программно (см. описание регистра SPMCR).

Напротив, код, расположенный в области NRWW, может быть

считан во время изменения страницы памяти программ, расположенной в области RWW. А при изменении содержимого области NRWW процессор останавливается до завершения операции. Сказанное выше проиллюстрировано на **Рис. 4.17**.

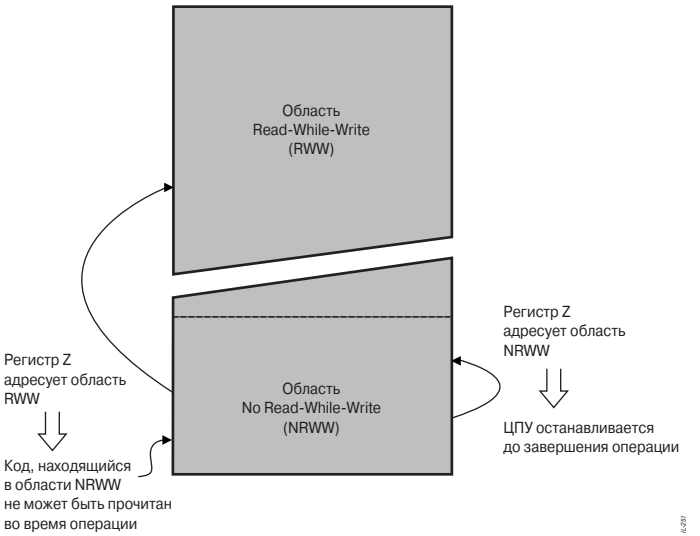


Рис. 4.17. Отличие между областями RWW и NRWW

В остальных моделях (ATmega161x, ATmega163x и ATmega323x) разделения на области RWW и NRWW нет. Однако условно можно считать, что секция прикладной программы является областью RWW, а секция загрузчика — областью NRWW соответственно.

26.3. Функционирование загрузчика

26.3.1. Управление процессом самопрограммирования

Изменение содержимого памяти программ и ячеек защиты загрузчика осуществляется с помощью команды SPM (Store Program Memory). Параметрами этой команды являются адрес области памяти, загружаемый предварительно в индексный регистр Z, и при необходимости данные, находящиеся в регистровой паре R1:R0.

Управление процессом программирования и, в частности, определение операции, выполняемой при вызове команды SPM, осуществля-

ется с помощью регистра ввода/вывода SPMCR (Store Program Memory Control Register). Во всех моделях, за исключением ATmega64x/128x, этот регистр расположен по адресу $37 ($57). В микроконтроллерах регистр ATmega64x/128x этот регистр расположен в пространстве дополнительных регистров ввода/вывода по адресу ($68). Формат этого регистра для различных моделей семейства показан на **Рис. 4.18**, а описание его разрядов приведено в **Табл. 4.26**.

Таблица 4.26. **Разряды регистра SPMCR**

Разряд		Описание	Примечание
7	SPMIE	**Разрешение прерывания SPM.** Если в этом разряде записана лог. 1 и флаг I регистра SREG также установлен в «1», то разрешается прерывание готовности SPM. Прерывание генерируется все время, пока разряд SPMEN регистра сброшен в «0».	Кроме ATmega161x ATmega163x ATmega323x
6	—	Зарезервировано, читается как «0»	ATmega161x
	ASB	**Запрещен доступ к секции прикладной программы.** Этот флаг показывает возможность обращения по адресам, расположенным в секции прикладной программы. Если флаг установлен в «1», доступ к этой секции запрещен, если сброшен в «0» — разрешен. Установка этого флага осуществляется аппаратно при выполнении операций записи или стирания страницы памяти. Сброс флага осуществляется либо программно, записью лог. 1 в разряд ASRE по окончании операции, либо аппаратно, при запуске операции загрузки страницы	ATmega163x ATmega323x
	RWWSB	**Запрещен доступ к области RWW.** Этот флаг показывает возможность обращения по адресам, расположенным в области RWW. Если флаг установлен в «1», доступ к области RWW запрещен, если сброшен в «0» — разрешен. Установка этого флага осуществляется аппаратно при выполнении операций записи или стирания страницы памяти. Сброс флага осуществляется либо программно, записью лог. 1 в разряд RWWSRE по окончании операции, либо аппаратно, при запуске операции загрузки страницы	ATmega8x ATmega8515x ATmega16x ATmega64x ATmega64x ATmega128x
5	—	Зарезервировано, читается как «0»	Все модели
4	—	Зарезервировано, читается как «0»	ATmega161x
	ASRE	**Чтение секции прикладной программы разрешено.** Одновременная установка этого разряда и разряда SPMEN позволяет разрешить доступ к секции прикладной программы. Разрешение доступа осуществляется командой SPM, запущенной в течение 4-х машинных циклов после установки указанных разрядов	ATmega163x ATmega323x
	RWWSRE	**Чтение области RWW разрешено.** Одновременная установка этого разряда и разряда SPMEN позволяет разрешить доступ к области RWW. Разрешение доступа к этой области осуществляется запуском команды SPM в течение 4-х машинных циклов после установки указанных разрядов. Разрешение доступа к области RWW может осуществляться только после завершения операции программирования (после сброса флага SPMEN)	ATmega8x ATmega8515x ATmega16x ATmega64x ATmega64x ATmega128x

Продолжение таблицы 4.26

Разряд		Описание	Примечание
3	BLBSET	**Изменение ячеек защиты загрузчика.** При одновременной установке этого разряда и разряда SPMEN, команда SPM, запущенная в течение 4-х машинных циклов, осуществит установку защитных ячеек загрузчика в соответствии с содержимым регистра R0. Сброс разряда BLBSET осуществляется аппаратно после установки ячеек защиты либо по истечении указанного времени. По команде LPM, запущенной в течение 3-х машинных циклов после установки указанных разрядов, будет осуществлено чтение либо конфигурационных ячеек, либо ячеек защиты (зависит от значения разряда Z0 регистра Z)	Все модели
2	PGWRT	**Запись страницы.** При одновременной установке этого разряда и разряда SPMEN, команда SPM, запущенная в течение 4-х машинных циклов, осуществит запись страницы памяти программ из временного буфера. Адрес страницы должен быть загружен в старший байт регистра Z (R31). Сброс разряда PGWRT осуществляется аппаратно по окончании записи страницы либо по истечении указанного времени. При записи в секцию NRWW центральный процессор останавливается на время выполнения операции*	Все модели
1	PGERS	**Стирание страницы.** При одновременной установке этого разряда и разряда SPMEN, команда SPM, запущенная в течение 4-х машинных циклов, осуществит стирание страницы памяти программ из временного буфера. Адрес страницы должен быть загружен в старший байт регистра Z (R31). Сброс разряда PGERS осуществляется аппаратно по окончании стирания страницы либо по истечении указанного времени. При записи в секцию NRWW центральный процессор останавливается на время выполнения операции*	Все модели
0	SPMEN	**Разрешение выполнения команды SPM.** Установка этого разряда разрешает запуск команды SPM в течение 4-х машинных циклов. Если разряд SPMEN устанавливается одновременно с одним из разрядов RWWSRE, BLBSET, PGWRT или PGERS, выполняется операция, определяемая этим разрядом (см. описание разрядов). Если устанавливается только разряд SPMEN, осуществляется сохранение содержимого регистров R1:R0 во временном буфере по адресу, находящемуся в регистре Z (M3P регистра игнорируется). Сброс разряда SPMEN осуществляется аппаратно после завершения операции либо по истечении указанного времени	Все модели
* В моделях ATmega161x, ATmega163x и ATmega323x останавливается при обращении по любому адресу памяти программ.			

Запись в младшие пять разрядов регистра значений, отличных от «10001», «01001», «00101», «00011» и «00001» не вызывает никакого эффекта.

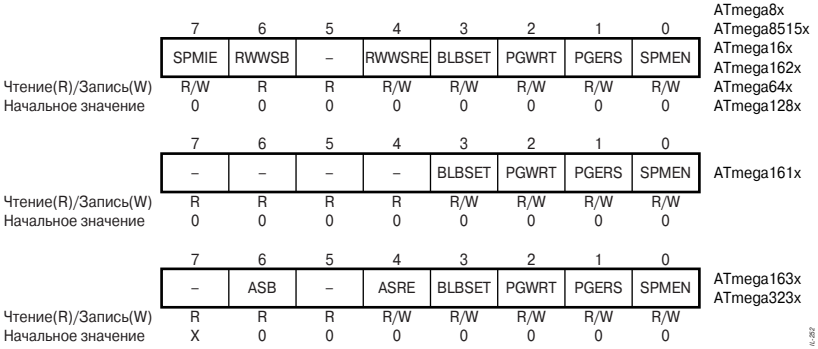
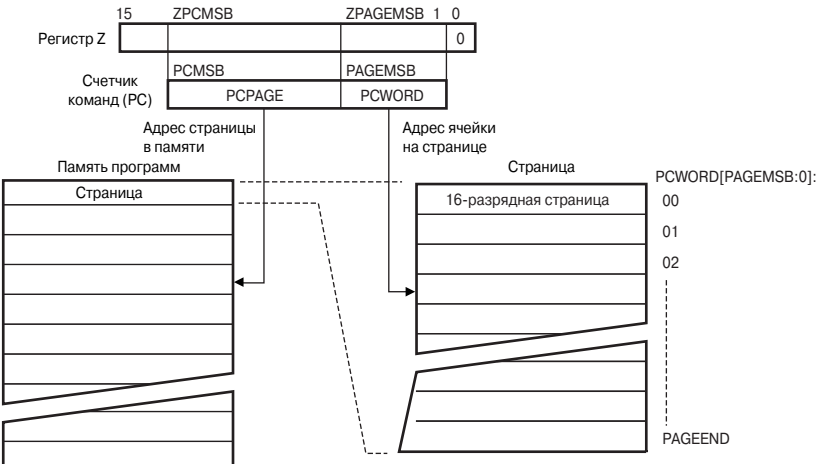


Рис. 4.18. Формат регистра SPMCR



Параметр	ATmega8x ATmega8515x	ATmega16x ATmega161x ATmega162x ATmega163x	ATmega32x ATmega323x	ATmega64x	ATmega128x
PCMSB	11	12	16	14	15
PAGEMSB	4	5	64	7	6
ZPCMSB	12	Z13	Z14	Z15	Z16*
ZPAGEMSB	5	Z6	Z7	Z7	Z7
PCPAGE	PC[11:5]	PC[12:6]	PC[13:6]	PC[14:7]	PC[15:7]
	Z12:Z6	Z13:Z7	Z14:Z7	Z15:Z8	Z16:Z8
PCWORD	PC[4:0]	PC[5:0]	PC[5:0]	PC[6:0]	PC[6:0]
	Z5:Z1	Z6:Z1	Z6:Z1	Z7:Z1	Z7:Z1

* Расположен в регистре RAMPZ.

Рис. 4.19. Адресация памяти программ при использовании команды SPM

Обратите внимание на то, что во время записи в EEPROM-память изменение регистра SPMCR невозможно. Поэтому, перед тем как записать какое-либо значение в регистр SPMCR, рекомендуется дождаться сброса флага EEWE регистра EECR.

Для адресации памяти программ при использовании команды SPM используется индексный регистр Z, получаемый объединением двух старших регистров общего назначения R30 (младший байт) и R31 (старший байт). Поскольку память программ в микроконтроллерах семейства Mega имеет страничную организацию, счетчик команд можно условно разбить на две части. Первая часть (младшие разряды) адресует ячейку на странице, а вторая часть определяет страницу (**Рис. 4.19**). После запуска операции программирования содержимое регистра Z фиксируется и его можно использовать для других целей.

26.3.2. Изменение памяти программ

Изменение содержимого памяти программ осуществляется в следующей последовательности:

1. Заполнение временного буфера страницы новым содержимым.
2. Очистка страницы.
3. Перенос содержимого буфера в память программ.

Следует заметить, что очистка страницы может выполняться как после заполнения буфера, так и перед его заполнением. Однако при необходимости изменить только часть страницы приведенный порядок действий является, по понятным причинам, единственно возможным. В этом случае содержимое ячеек, не требующих изменения, сохраняется в буфере перед очисткой страницы.

Для определения момента окончания выполнения операций можно либо опрашивать состояние флага SPMEN регистра SPMCR, дожидаясь его сброса, либо воспользоваться прерыванием «Готовность SPM». Это прерывание генерируется все время, пока флаг SPMEN сброшен. В последнем случае таблица векторов прерываний должна находиться в секции загрузчика, а это прерывание должно быть разрешено установкой флага SPMIE регистра SPMCR.

Для стирания страницы памяти программ необходимо занести адрес страницы в регистр Z (секция PCPAGE), записать значение «x0000011» в регистр SPMCR и в течение четырех машинных циклов выполнить команду SPM. Содержимое регистров R1 и R0 при этом игнорируется.

Для занесения слова команды в буфер следует загрузить адрес ячейки в регистр Z (секция PCWORD), а код операции — в регистры R1:R0. После этого необходимо записать значение «x0000011» в регистр

SPMCR и в течение четырех машинных циклов, выполнить команду SPM. Очистка буфера осуществляется автоматически по окончании записи страницы либо вручную, записью лог. 1 в разряд RWWSRE регистра SPMCR. Заметим, что запись по одному и тому же адресу в буфере невозможна без его очистки.

Запись содержимого буфера в память программ осуществляется аналогично. В регистр Z (секция PCPAGE) заносится адрес страницы, в регистр SPMCR записывается значение «x0000101» и в течение четырех машинных циклов выполняется команда SPM. Содержимое регистров R1 и R0 при этом игнорируется.

26.3.3. Изменение ячеек защиты загрузчика

Изменение ячеек защиты загрузчика BLB12:BLB11 и BLB02:BLB01 также осуществляется командой SPM. Для этого необходимо загрузить в регистр R0 требуемое значение в соответствии с **Рис. 4.20** (сброшенный разряд означает программирование соответствующей ячейки).

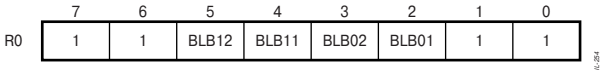


Рис. 4.20. Изменение состояния ячеек защиты регистра RO по команде SPM

После этого необходимо записать значение «x0001001» в регистр SPMCR и в течение четырех машинных циклов выполнить команду SPM. Содержимое регистра Z при этом игнорируется, однако для совместимости с будущими устройствами рекомендуется записывать в него значение $0001. Во время программирования ячеек защиты можно обращаться к любой области памяти программ.

26.3.4. Чтение конфигурационных ячеек и ячеек защиты

Помимо программирования микроконтроллера, загрузчик может также считывать содержимое конфигурационных ячеек и ячеек защиты. Так, для чтения байта защиты следует загрузить в регистр Z число $0001, записать в регистр SPMCR значение «x0001001» и в течение трех машинных циклов выполнить команду LPM. В результате содержимое байта защиты будет занесено в заданный регистр общего назначения. Соответствие разрядов регистра ячейкам приведено на **Рис. 4.1**.

Чтение конфигурационных байтов осуществляется аналогично. В регистр Z загружается адрес байта ($0000 — младший байт, $0003 — старший байт, $0002 — дополнительный байт), после чего необходимо

записать в регистр SPMCR значение «x0001001» и в течение трех машинных циклов выполнить команду LPM. В результате выполнения команды содержимое выбранного байта конфигурации будет занесено в регистр общего назначения. Соответствие разрядов регистра конфигурационным ячейкам — см. **Табл. 4.6**.

26.3.5. Пример реализации программы-загрузчика

Ниже приведен пример реализации программы-загрузчика, осуществляющей простое копирование страницы памяти программ из ОЗУ в FLASH-память. Этот пример предназначен исключительно для иллюстрации возможностей самопрограммирования микроконтроллеров семейства, поэтому в нем отсутствуют некоторые элементы, обязательные в реальной программе. В частности, отсутствует такая важная вещь, как обработка ошибок. Кроме того, предполагается, что прерывания запрещены или что таблица векторов размещена в секции загрузчика.

```
;Адрес 1-го байта в ОЗУ передается в указателе Y
;Адрес 1-го байта в FLASH-памяти передается в указателе Z
.equ PAGESIZEB = PAGESIZE*2  ;Размер страницы в байтах
.org SMALLBOOTSTART
Write_page:
;Стереть страницу
ldi spmcrval,(1<<PGERS)|(1<<SPMEN)
call Do_spm

;Разрешить адресацию области RWW
ldi spmcrval,(1<<RWWSRE)|(1<<SPMEN)
call Do_spm

;Передать данные из ОЗУ в буфер страницы
ldi looplo,low(PAGESIZEB)    ;Инициализировали счетчик байтов
ldi loophi,high(PAGESIZEB)
Wrloop:
ld r0,Y+
ld r1,Y+
ldi spmcrval,(1<<SPMEN)
call Do_spm
adiw ZH:ZL,2
sbiw loophi:looplo,2
brne Wrloop
```

```
;Записать страницу
subi ZL,low(PAGESIZEB)       ;Восстановили указатель
subci ZH,high(PAGESIZEB)
ldi spmcrval,(1<<PGWRT)|(1<<SPMEN)
call Do_spm

;Разрешить адресацию области RWW
ldi spmcrval,(1<<RWWSRE)|(1<<SPMEN)
call Do_spm

;Проконтролировать записанные данные
ldi looplo,low(PAGESIZEB)    ;Инициализировали счетчик байтов
ldi loophi,high(PAGESIZEB)
subi YL,low(PAGESIZEB)       ;Восстановили указатель
subci YH,high(PAGESIZEB)
Rdloop:
lpm r0,Z+
ld r1,Y+
cpse r0,r1
jmp Error
sbiw loophi:looplo,1
brne Rdloop

;Возврат в секцию прикладной программы
Return:
lds temp1,SPMCR
sbrs temp1,RWWSB             ;Если RWWSB установлен, доступ в
                             ;секцию RWW запрещен
ret
;Разрешить адресацию области RWW
ldi spmcrval,(1<<RWWSRE)|(1<<SPMEN)
call Do_spm
rjmp Return

Do_spm:                      ;Операция определяется содержимым
                             ;spmcrval. Проверить завершение
                             ;предыдущей операции
Wait_spm:
lds temp1,SPMCR
sbrc temp1,SPMEN
rjmp Wait_spm
```

```
;Адрес 1-го байта в ОЗУ передается в указателе Y
;Адрес 1-го байта в FLASH-памяти передается в указателе Z
.equ PAGESIZEB = PAGESIZE*2  ; размер страницы в байтах
.org SMALLBOOTSTART
Write_page:
;Стереть страницу
ldi spmcrval,(1<<PGERS)|(1<<SPMEN)
call Do_spm

;Разрешить адресацию области RWW
ldi spmcrval,(1<<RWWSRE)|(1<<SPMEN)
call Do_spm

;Передать данные из ОЗУ в буфер страницы
ldi looplo,low(PAGESIZEB)    ;Инициализировали счетчик байтов
ldi loophi,high(PAGESIZEB)
Wrloop:
ld r0,Y+
ld r1,Y+
ldi spmcrval,(1<<SPMEN)
call Do_spm
adiw ZH:ZL,2
sbiw loophi:looplo,2
brne Wrloop

;Записать страницу
subi ZL,low(PAGESIZEB)       ;Восстановили указатель
subci ZH,high(PAGESIZEB)
ldi spmcrval,(1<<PGWRT)|(1<<SPMEN)
call Do_spm

;Разрешить адресацию области RWW
ldi spmcrval,(1<<RWWSRE)|(1<<SPMEN)
call Do_spm

;Проконтролировать записанные данные
ldi looplo,low(PAGESIZEB)    ;Инициализировали счетчик байтов
ldi loophi,high(PAGESIZEB)
subi YL,low(PAGESIZEB)       ;Восстановили указатель
subci YH,high(PAGESIZEB)
Rdloop:
```

```
lpm r0,Z+
ld r1,Y+
cpse r0,r1
jmp Error
sbiw loophi:looplo,1
brne Rdloop

;Возврат в секцию прикладной программы
Return:
lds temp1,SPMCR
sbrs temp1,RWWSB            ;Если RWWSB установлен, доступ в
                            ;секцию RWW запрещен
ret
;Разрешить адресацию области RWW
ldi spmcrval,(1<<RWWSRE)|(1<<SPMEN)
call Do_spm
rjmp Return

Do_spm:                     ;Операция определяется содержимым
                            ;spmcrval. Проверить завершение
                            ;предыдущей операции
Wait_spm:
lds temp1,SPMCR
sbrc temp1,SPMEN
rjmp Wait_spm
;Запретить прерывания, сохранить регистр статуса
in temp2,SREG
cli
;Убедиться в отсутствии записи в EEPROM
Wait_ee:
sbic EECR,EEWE
rjmp Wait_ee

sts SPMCR,spmcrval
spm
;Восстановить регистр SREG (для повторного разрешения прерываний)
out SREG,temp2
ret
```

Приложения

1 Сводная таблица микроконтроллеров AVR семейства Tiny

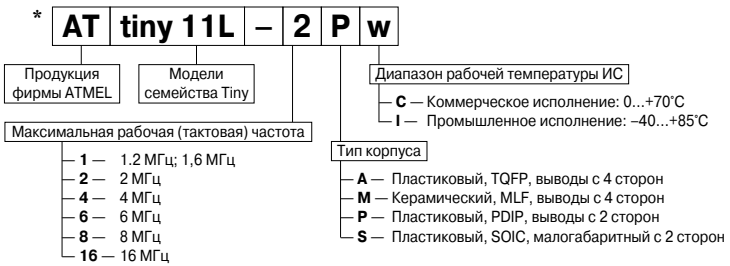
2 Сводная таблица микроконтроллеров AVR семейства Mega

3 Чертежи корпусов микроконтроллеров AVR семейств Tiny и Mega

4 Электрические параметры микроконтроллеров AVR семейств Tiny и Mega

Приложение 1. Сводная таблица микроконтроллеров

Обозначение*	Память программ (FLASH) [Кбайт]	Память данных [байт]		Кол-во команд	Кол-во линий ввода/вывода	Кол-во источников прерываний		Таймеры	
		EEPROM	SRAM			внутр.	внешн.	8-разрядные	сторожевой
ATtiny11L–2Pw	1	—	—	90	6	3	1	1	●
ATtiny11L–2Sw									
ATtiny11–6Pw	1	—	—	90	6	3	1	1	●
ATtiny11–6Sw									
ATtiny12V–1Pw	1	64	—	90	6	4	1	1	●
ATtiny12V–1Sw									
ATtiny12L–4Pw	1	64	—	90	6	4	1	1	●
ATtiny12L–4Sw									
ATtiny12–8Pw	1	64	—	90	6	4	1	1	●
ATtiny12–8Sw									
ATtiny15L–1Pw	1	64	—	90	6	7	1	2	●
ATtiny15L–1Sw									
ATtiny26L–8Pw	2	128	128	118	16	9	2	2	●
ATtiny26L–8Sw									
ATtiny26L–8Mw									
ATtiny26–16Pw	2	128	128	118	16	9	2	2	●
ATtiny26–16Sw									
ATtiny26–16Mw									
ATtiny28V–1Pw	2	—	—	90	19	3	2	1	●
ATtiny28V–1Aw									
ATtiny28V–1Mw									
ATtiny28L–4Pw	2	—	—	90	19	3	2	1	●
ATtiny28L–4Aw									
ATtiny28L–4Mw									



AVR семейства Tiny

ШИМ	Аналоговый компаратор	АЦП (10-разр.) [каналов]	Встроенный *RC*-генератор	Схема BOD	Возможность внутрисхемного программирования	Рабочая частота [МГц]	Напряжение питания [В]	Корпус	
								Тип	№ рис. (Прил. 3)
—	●	—	●	—	—	0...2	2.7...5.5	DIP-8	1
								SOIC-8	5
—	●	—	●	—	—	0...6	4.0...5.5	DIP-8	1
								SOIC-8	5
—	●	—	●	●	●	0...1.2	1.8...5.5	DIP-8	1
								SOIC-8	5
—	●	—	●	●	●	0...4	2.7...5.5	DIP-8	1
								SOIC-8	5
—	●	—	●	●	●	0...8	4.0...5.5	DIP-8	1
								SOIC-8	5
1	●	4	●	●	●	1.6	2.7...5.5	DIP-8	1
								SOIC-8	5
2	●	11/(8 + 7)	●	●	●	0...8	2.7...5.5	DIP-20	2
								SOIC-20	6
								MLF-32	10
2	●	11/(8 + 7)	●	●	●	0...16	4.5...5.5	DIP-20	2
								SOIC-20	6
								MLF-32	10
—	●	—	●	—	●	0...1.2	2.7...6.0	DIP-28	3
								TQFP-32	7
								MLF-32	10
—	●	—	●	—	●	0...4	4.0...6.0	DIP-28	3
								TQFP-32	7
								MLF-32	10

Приложение 2. Сводная таблица микроконтроллеров

Обозначение*	Память программ (FLASH) [Кбайт]	Память данных [байт]		Внешнее ОЗУ	Кол-во команд	Кол-во линий ввода/вывода	Кол-во источников прерываний		Таймеры			ШИМ
		EEPROM	SRAM				внутр.	внешн.	8-разрядные	16-разрядные	сторожевой	
ATmega8L–8Pw	8	512	1K	—	130	23	16	2	2	1	●	3
ATmega8L–8Aw												
ATmega8L–8Mw												
ATmega8–16Pw	8	512	1K	—	130	23	16	2	2	1	●	3
ATmega8–16Aw												
ATmega8–16Mw												
ATmega8515L–8Pw	8	512	512	●	130	35	13	3	1	1	●	3
ATmega8515L–8Aw												
ATmega8515L–8Mw												
ATmega8515L–8Jw												
ATmega8515–16Pw	8	512	512	●	130	35	13	3	1	1	●	3
ATmega8515–16Aw												
ATmega8515–16Mw												
ATmega8515–16Jw												
ATmega8535L–8Pw	8	512	512	—	130	32	17	3	2	1	●	4
ATmega8535L–8Aw												
ATmega8535L–8Mw												
ATmega8535L–8Jw												
ATmega8535L–16Pw	8	512	512	—	130	32	17	3	2	1	●	4
ATmega8535L–16Aw												
ATmega8535L–16Mw												
ATmega8535L–16Jw												
ATmega16L–8Pw	16	512	1K	—	130	32	17	3	2	1	●	4
ATmega16L–8Aw												
ATmega16–16Pw	16	512	1K	—	130	32	17	3	2	1	●	4
ATmega16–16Aw												
ATmega161–4Pw	16	512	1K	●	130	35	17	3	2	1	●	4
ATmega161–4Aw												
ATmega161–8Pw	16	512	1K	●	130	35	17	3	2	1	●	4
ATmega161–8Aw												
ATmega162V–1Pw	16	512	1K	●	130	35	24	3	2	2	●	6
ATmega162V–1Aw												
ATmega162V–1Mw												

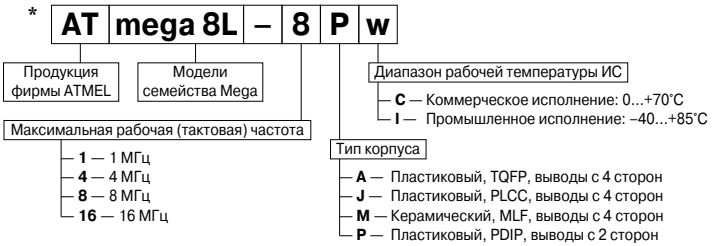
AVR семейства Mega

Интерфейсы				Аналоговый компаратор	АЦП (10-разр.) [каналов]	Встроенный *RC*-генератор	Схема BOD	Возможность внутрисхемного программирования	Рабочая частота [МГц]	Напряжение питания [В]	Корпус	
SPI	TWI	UART	USART								Тип	№ рис. (Прил. 3)
1	1	—	1	●	4+2	●	●	●	0...8	2.7...5.5	DIP-28	3
					6+2						TQFP-32	7
					6+2						MLF-32	10
1	1	—	1	●	4+2	●	●	●	0...16	4.5...5.5	DIP-28	3
					6+2						TQFP-32	7
					6+2						MLF-32	10
1	—	—	1	●	—	●	●	●	0...8	2.7...5.5	DIP-28	3
											TQFP-32	7
											MLF-32	10
											PLCC-44	13
1	—	—	1	●	—	●	●	●	0...16	4.5...5.5	DIP-28	3
											TQFP-32	7
											MLF-32	10
											PLCC-44	13
1	1	—	1	●	8	●	●	●	0...8	2.7...5.5	DIP-40	4
					8/(7 + 2)						TQFP-44	8
											MLF-44	11
											PLCC-44	13
1	1	—	1	●	8	●	●	●	0...16	4.5...5.5	DIP-40	4
					8/(7 + 2)						TQFP-44	8
											MLF-44	11
											PLCC-44	13
1	1	—	1	●	8	●	●	●	0...8	2.7...5.5	DIP-40	4
					8/(7 + 2)						TQFP-44	8
1	1	—	1		8	●	●	●	0...16	4.5...5.5	DIP-40	4
					8/(7 + 2)						TQFP-44	8
1	—	2	—	●	—	—	●	●	0...4	2.7...5.5	DIP-40	4
											TQFP-44	8
1	—	2	—	●	—	—	●	●	0...8	4.5...5.5	DIP-40	4
											TQFP-44	8
1	—	—	2	●	—	●	●	●	0...1	1.8...3.6	DIP-40	4
											TQFP-44	8
											MLF-44	11

Обозначение*	Память программ (FLASH) [Кбайт]	Память данных [байт]		Внешнее ОЗУ	Кол-во команд	Кол-во линий ввода/вывода	Кол-во источников прерываний		Таймеры			ШИМ
		EEPROM	SRAM				внутр.	внешн.	8-разрядные	16-разрядные	сторожевой	
ATmega162L–8Pw	16	512	1K	●	130	35	24	3	2	2	●	6
ATmega162L–8Aw												
ATmega162L–8Mw												
ATmega162–16Pw	16	512	1K	●	130	35	24	3	2	2	●	6
ATmega162–16Aw												
ATmega162–16Mw												
ATmega163L–4Pw	16	512	1K	—	130	32	15	2	2	1	●	3
ATmega163L–4Aw												
ATmega163–8Pw	16	512	1K	—	130	32	15	2	2	1	●	3
ATmega163–8Aw												
ATmega169V–1Aw	16	512	1K	—	130	53	19	3	2	1	●	4
ATmega169V–1Mw												
ATmega169L–8Aw	16	512	1K	—	130	53	19	3	2	1	●	4
ATmega169L–8Mw												
ATmega169–16Aw	16	512	1K	—	130	53	19	3	2	1	●	4
ATmega169–16Mw												
ATmega162V–1Pw	32	1K	2K	—	130	32	16	3	2	1	●	4
ATmega162V–1Aw												
ATmega162V–1Mw												
ATmega162V–1Pw	32	1K	2K	—	130	32	16	3	2	1	●	4
ATmega162V–1Aw												
ATmega162V–1Mw												
ATmega323L–4Pw	32	1K	2K	—	130	32	16	3	2	1	●	4
ATmega323L–4Aw												
ATmega323–8Pw	32	1K	2K	—	130	32	16	3	2	1	●	4
ATmega323–8Aw												
ATmega64L–8Aw	64	2K	4K	●	130	53	26	8	2	2	●	6 + 2
ATmega64–16Aw												
ATmega323L–4Pw	32	1K	2K	●	133	53	26	8	2	2	●	6 + 2
ATmega323L–4Aw												

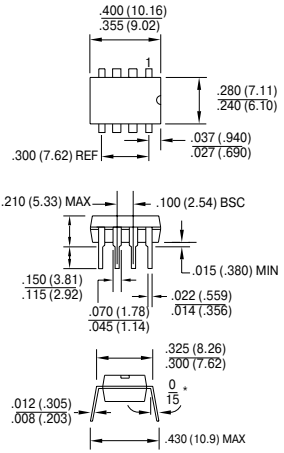
Продолжение

Интерфейсы				Аналоговый компаратор	АЦП (10-разр.) [каналов]	Встроенный *RC*-генератор	Схема BOD	Возможность внутрисхемного программирования	Рабочая частота [МГц]	Напряжение питания [В]	Корпус	
SPI	TWI	UART	USART								Тип	№ рис. (Прил. 3)
1	—	—	2	●	—	●	●	●	0...8	2.7...5.5	DIP-40	4
											TQFP-44	8
											MLF-44	11
1	—	—	2	●	—	●	●	●	0...16	4.5...5.5	DIP-40	4
											TQFP-44	8
											MLF-44	11
1	1	1	—	●	8	●	●	●	0...4	2.7...5.5	DIP-40	4
											TQFP-44	8
1	1	1	—	●	8	●	●	●	0...8	4.0...5.5	DIP-40	4
											TQFP-44	8
1	1(USI)	—	—1	●	8	—●	●	●	0...1	1.8...5.5	TQFP-64	9
											MLF-64	12
1	1(USI)	—	1	●	8	●	●	●	0...8	2.7...5.5	TQFP-64	9
											MLF-64	12
1	1(USI)	—	1	●	8	●	●	●	0...16	4.5...5.5	TQFP-64	9
											MLF-64	12
1	1	—	1	●	8	●	●	●	0...8	2.7...5.5	DIP-40	4
					8/(7 + 2)						TQFP-44	8
											MLF-44	11
1	1	—	1	●	8	●	●	●	0...16	4.5...5.5	DIP-40	4
					8/(7 + 2)						TQFP-44	8
											MLF-44	11
1	1	—	1	●	8	●	●	●	0...4	2.7...5.5	DIP-40	4
											TQFP-44	8
1	1	—	1	●	8	●	●	●	0...8	4.0...5.5	DIP-40	4
											TQFP-44	8
1	1	—	2	●	8/(7 + 2)	●	●	●	0...8	2.7...5.5	TQFP-64	9
									0...16	4.5...5.5		
1	1	—	2	●	8/(7 + 2)	●	●	●	0...8	2.7...5.5	TQFP-64	9
									0...16	4.5...5.5		


* AT mega 8L – 8 P w
Продукция фирмы ATMEL
Модели семейства Mega
Максимальная рабочая (тактовая) частота
1 — 1 МГц
4 — 4 МГц
8 — 8 МГц
16 — 16 МГц
Тип корпуса
A — Пластиковый, TQFP, выводы с 4 сторон
J — Пластиковый, PLCC, выводы с 4 сторон
M — Керамический, MLF, выводы с 4 сторон
P — Пластиковый, PDIP, выводы с 2 сторон
Диапазон рабочей температуры ИС
C — Коммерческое исполнение: 0...+70°C
I — Промышленное исполнение: –40...+85°C

Приложение 3. Чертежи корпусов микроконтроллеров AVR семейств Tiny и Mega

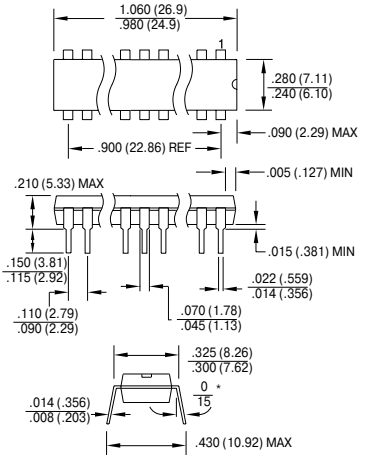
8-выводный пластиковый корпус с двурядным расположением выводов (PDIP-8)



* Размер для справок

1 Размеры в дюймах и миллиметрах

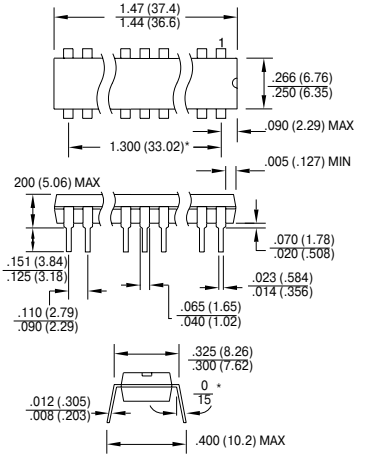
20-выводный пластиковый корпус с двурядным расположением выводов (PDIP-20)



* Размер для справок

2 Размеры в дюймах и миллиметрах

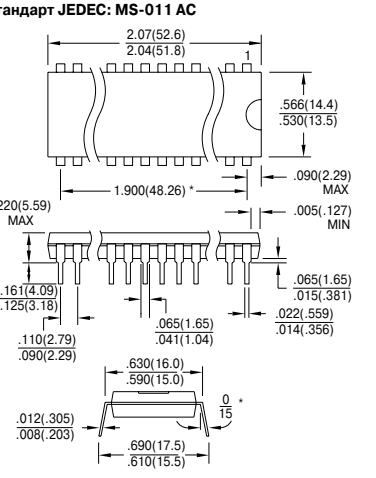
28-выводный пластиковый корпус с двурядным расположением выводов (PDIP-28)



* Размер для справок

3 Размеры в дюймах и миллиметрах

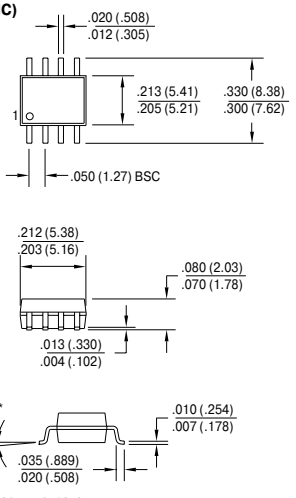
40-выводный пластиковый корпус с двурядным расположением выводов (PDIP-40)
Стандарт JEDEC: MS-011 AC



* Размер для справок

4 Размеры в дюймах и миллиметрах

8-выводный SOIC корпус, ширина 0.20 in (EIAJ SOIC)

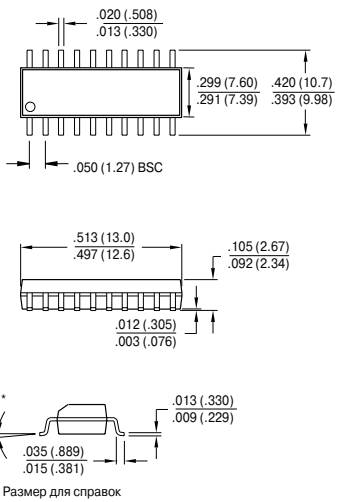


* Размер для справок

Размеры в дюймах и миллиметрах

5

20-выводный SOIC корпус, ширина 0.300 in

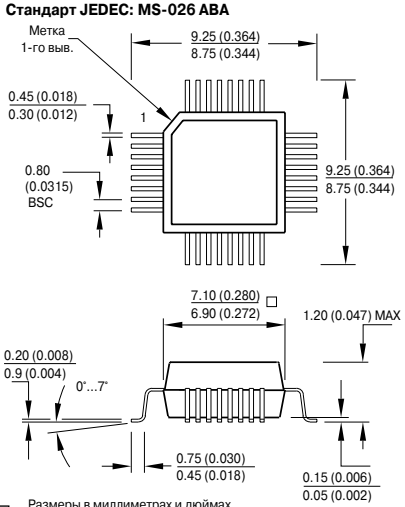


* Размер для справок

Размеры в дюймах и миллиметрах

6

38-выводный TQFP корпус (7×7 мм, шаг выводов — 0.8 мм) Стандарт JEDEC: MS-026 ABA

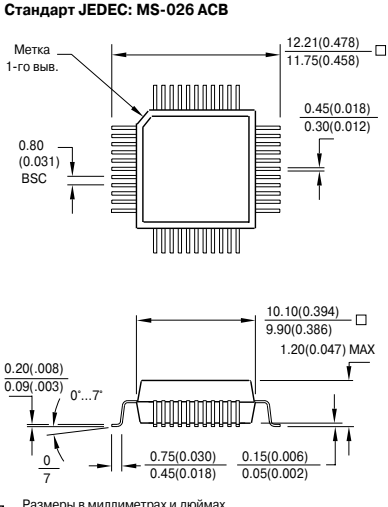


Размеры в миллиметрах и дюймах

7

44-выводный TQFP корпус (толщина 1.0 мм)

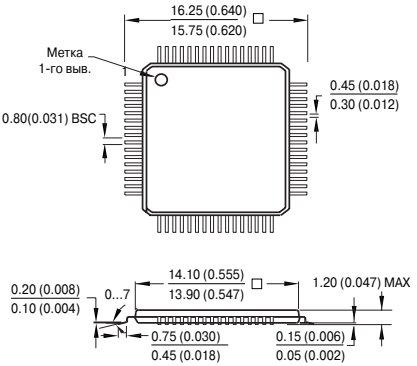
Стандарт JEDEC: MS-026 ACB



Размеры в миллиметрах и дюймах

8

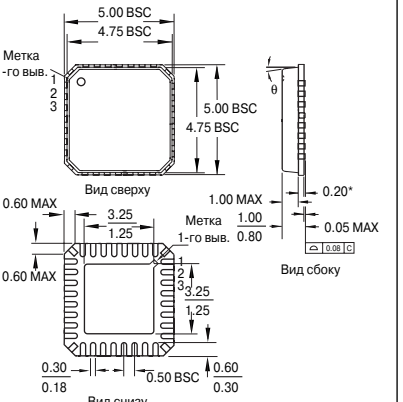
64-выводный TQFP корпус
(14×14 мм, шаг выводов — 0.8 мм)
Стандарт JEDEC: MS-026 AEB



Контролируемая размерность — миллиметры
Размеры в миллиметрах и дюймах

9

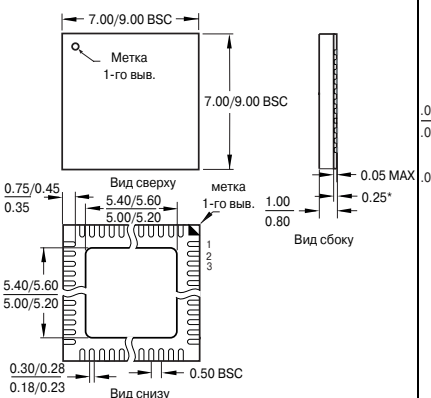
32-выводный MLF корпус
(5×5×1 мм, шаг выводов — 0.5 мм)
Стандарт JEDEC: MO-220



* Размер для справок
Размеры в миллиметрах

10

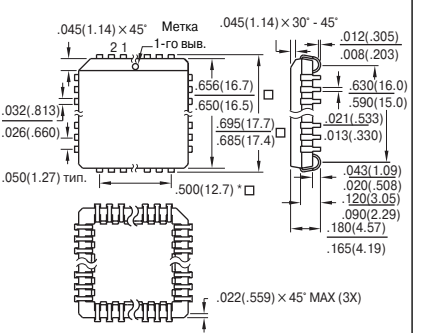
44/64-выводный MLF корпус
(7×7×1/9×9×1 мм, шаг выводов — 0.5 мм)
Стандарт JEDEC: MO-220



* Размер для справок
Размеры в миллиметрах

11/12

44-выводный PLCC корпус
Стандарт JEDEC: MS-018 AC



* Размер для справок
Размеры в дюймах и миллиметрах

13

Приложение 4. Электрические параметры микроконтроллеров AVR семейств Tiny и Mega

Максимально допустимые значения

Параметр	Предельные значения
Рабочая температура	–40...+85/105°C (модели ATtiny28x) –55...+125°C (остальные модели)
Температура хранения	–65...+150°C
Напряжение на любом выводе (кроме вывода $\overline{\text{RESET}}$) относительно вывода GND	–1.0...V_{CC} + 0.5 В
Напряжение на выводе $\overline{\text{RESET}}$ относительно вывода GND	–1.0...+13 В
Напряжение питания	6.0 В
Максимальный ток канала ввода/вывода	40 мА 60 мА (модели ATtiny28x, вывод PA2)
Максимальный ток выводов V_{CC} и GND	100 мА (модели ATtiny15L) 300 мА (модели ATtiny28x) 200 мА (остальные модели)

Статические параметры (DC)

Параметр		Условия	Модель	min	typ	max	Ед. изм.
V_{IL}	Входное напряжение низкого уровня	Вывод XTAL1	Все модели	–0.5		$0.1V_{CC}$	В
		Остальные выводы	Все модели	–0.5		$0.3V_{CC}$	В
V_{IH}	Входное напряжение высокого уровня	Вывод XTAL1	ATtiny	$0.7V_{CC}$		V_{CC} + 0.5	В
			ATmega	$0.8V_{CC}$		V_{CC} + 0.5	
		Вывод $\overline{\text{RESET}}$	ATtiny	$0.85V_{CC}$		V_{CC} + 0.5	В
			ATmega	$0.9V_{CC}$		V_{CC} + 0.5	
		Остальные выводы	Все модели	$0.6V_{CC}$		V_{CC} + 0.5	В
V_{OL}	Выходное напряжение низкого уровня линий портов ввода/вывода	I_{OL} = 20 мА, V_{CC} = 5 В	Все модели			0.6	В
		I_{OL} = 10 мА, V_{CC} = 3 В	Все модели			0.5	
V_{OH}	Выходное напряжение высокого уровня линий портов ввода/вывода	I_{OH} = –3 мА, V_{CC} = 5 В	ATtiny, ATmega161/163/323	4.2			В
		I_{OH} = –20 мА, V_{CC} = 5 В	Остальные				
		I_{OH} = –1.5 мА, V_{CC} = 3 В	ATtiny, ATmega161/163/323	2.3			
		I_{OH} = –10 мА, V_{CC} = 3 В	Остальные				
		I_{OH} = –1 мА, V_{CC} = 2.5 В	AT90C8534	1.44			

Продолжение

Параметр		Условия	Модель	min	typ	max	Ед. изм.
I_{IL}	Ток утечки на входе	V_{CC} = 5.5 В, на выводе – лог. «0» (абсолютное значение)	ATmega8515x			3	мкА
			Остальные			8	
I_{IH}	Ток утечки на входе	V_{CC} = 5.5 В, на выводе – лог. «1» (абсолютное значение)	ATmega8x			3	мкА
			Остальные			8	
RRST	Сопротивление подтягивающего резистора в цепи сброса		Все (кроме ATtiny)	100		500	кОм
$R_{I/O}$	Сопротивление подтягивающего резистора линии порта ввода/вывода		Все	35		120	кОм
sI_{CC}	Ток потребления, семейство Tiny	Рабочий режим, V_{CC} = 3 В, $f \leq 4$ МГц	Все модели			3.0	мА
		Рабочий режим, V_{CC} = 5 В, $f > 4$ МГц	ATiny11x/12x			10	
		Режим «Idle», V_{CC} = 3 В, $f \leq 4$ МГц	Все модели			1.2	мА
		Режим «Idle», V_{CC} = 5 В, $f > 4$ МГц	ATiny11x/12x			3.5	
		Режим «Power Down» WDT – вкл., V_{CC} = 3 В	Все модели		9.0	15.0	мкА
		Режим «Power Down» WDT – выкл., V_{CC} = 3 В	Все модели		<1.0	2.0	
	Ток потребления, семейство Mega	Рабочий режим, V_{CC} = 3 В, f = 4 МГц	Все модели			6.0	мА
		Рабочий режим, V_{CC} = 5 В, f = 8 МГц	Все модели			15.0	
		Режим «Idle», V_{CC} = 3 В, f = 4 МГц	Все модели			2.5	мА
		Режим «Idle», V_{CC} = 5 В, f = 8 МГц	Все модели (кроме ATmega161x)			8.0	
		Режим «Power Down» WDT – вкл., V_{CC} = 3 В	Все модели		<1.0	30.0	мкА
		Режим «Power Down» WDT – выкл., V_{CC} = 3 В	Все модели			8.0	

Предметный указатель

P

R

S

T

U

W

X

А

В

И

К

О

П

Р

С

Т

Ц