Особенности

• 8-разрядный высокопроизводительный AVR микроконтроллер с малым потреблением
• Прогрессивная RISC архитектура
  130 высокопроизводительных команд, большинство команд выполняется за один тактовый цикл
  32 8-разрядных рабочих регистра общего назначения Полностью статическая работа
  Приближающаяся к 16 MIPS (при тактовой частоте 16 МГц) производительность
  Встроенный 2-цикловый перемножитель
• Энергонезависимая память программ и данных
  8 Кбайт внутрисистемно программируемой Flash памяти (In-System Self-Programmable Flash)
    Обеспечивает 1000 циклов стирания/записи
  Дополнительный сектор загрузочных кодов с независимыми битами блокировки
    Обеспечен режим одновременного чтения/записи (Read-While-Write)
  512 байт EEPROM
    Обеспечивает 100000 циклов стирания/записи
  1 Кбайт встроенной SRAM
  Программируемая блокировка, обеспечивающая защиту программных средств пользователя
• Встроенная периферия
  Два 8-разрядных таймера/счетчика с отдельным предварительным делителем, один с режимом сравнения
  Один 16-разрядный таймер/счетчик с отдельным предварительным делителем и режимами захвата и сравнения
  Счетчик реального времени с отдельным генератором
  Три канала PWM
  8-канальный аналого-цифровой преобразователь (в корпусах TQFP и MLF)
    6 каналов с 10-разрядной точностью
    2 канала с 8-разрядной точностью
  6-канальный аналого-цифровой преобразователь (в корпусе PDIP)
    4 канала с 10-разрядной точностью
    2 канала с 8-разрядной точностью
  Байт-ориентированный 2-проводный последовательный интерфейс
  Программируемый последовательный USART
  Последовательный интерфейс SPI (ведущий/ведомый)
  Программируемый сторожевой таймер с отдельным встроенным генератором
  Встроенный аналоговый компаратор
• Специальные микроконтроллерные функции
  Сброс по подаче питания и программируемый детектор кратковременного снижения напряжения питания
  Встроенный калиброванный RC-генератор
  Внутренние и внешние источники прерываний
  Пять режимов пониженного потребления: Idle, Power-save, Power-down, Standby и снижения шумов ADC
• Выводы I/O и корпуса
  23 программируемые линии ввода/вывода
  28-выводной корпус PDIP, 32-выводной корпус TQFP и 32-выводной корпус MLF
• Рабочие напряжения
  2,7 - 5,5 В (ATmega8L)
  4,5 - 5,5 В (ATmega8)
• Рабочая частота
  0 - 8 МГц (ATmega8L)
  0 - 16 МГц (ATmega8**)**

ATMEL

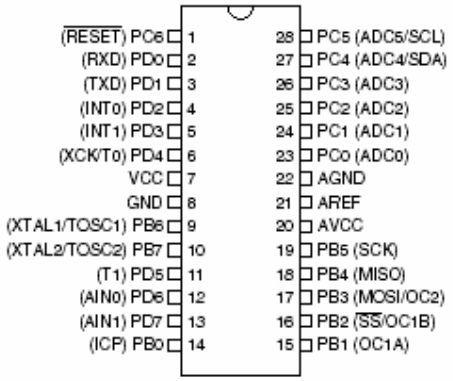
8-bit AVR® with 8K Bytes In-System Programmable Flash

ATmega8
ATmega8L

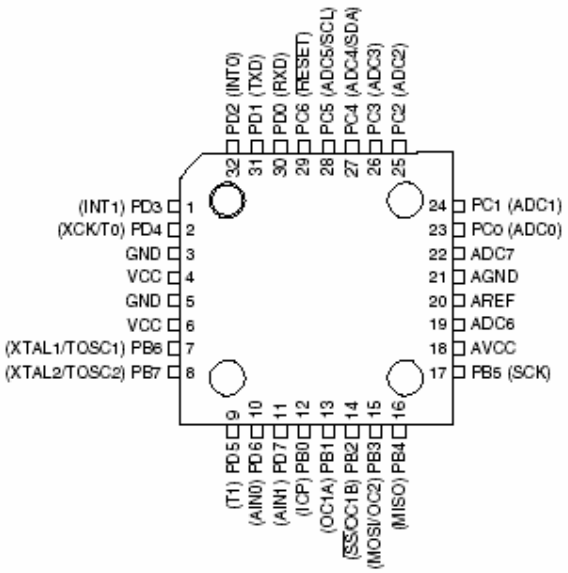
2456Q-AVR-10/06

Расположение выводов:

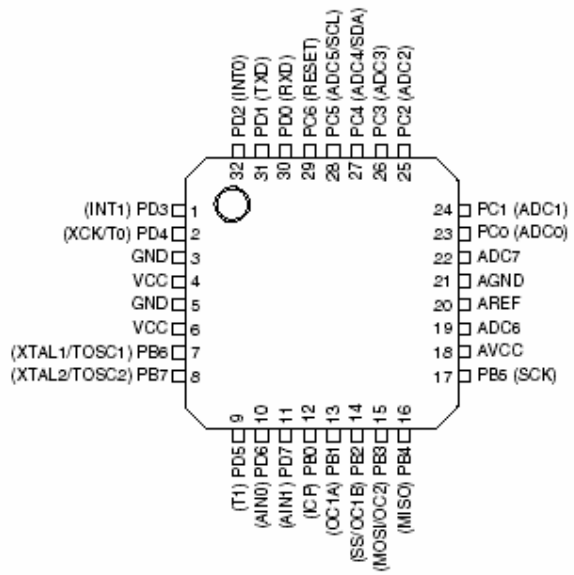
PDIP



TQFP Top View



MLF Top View

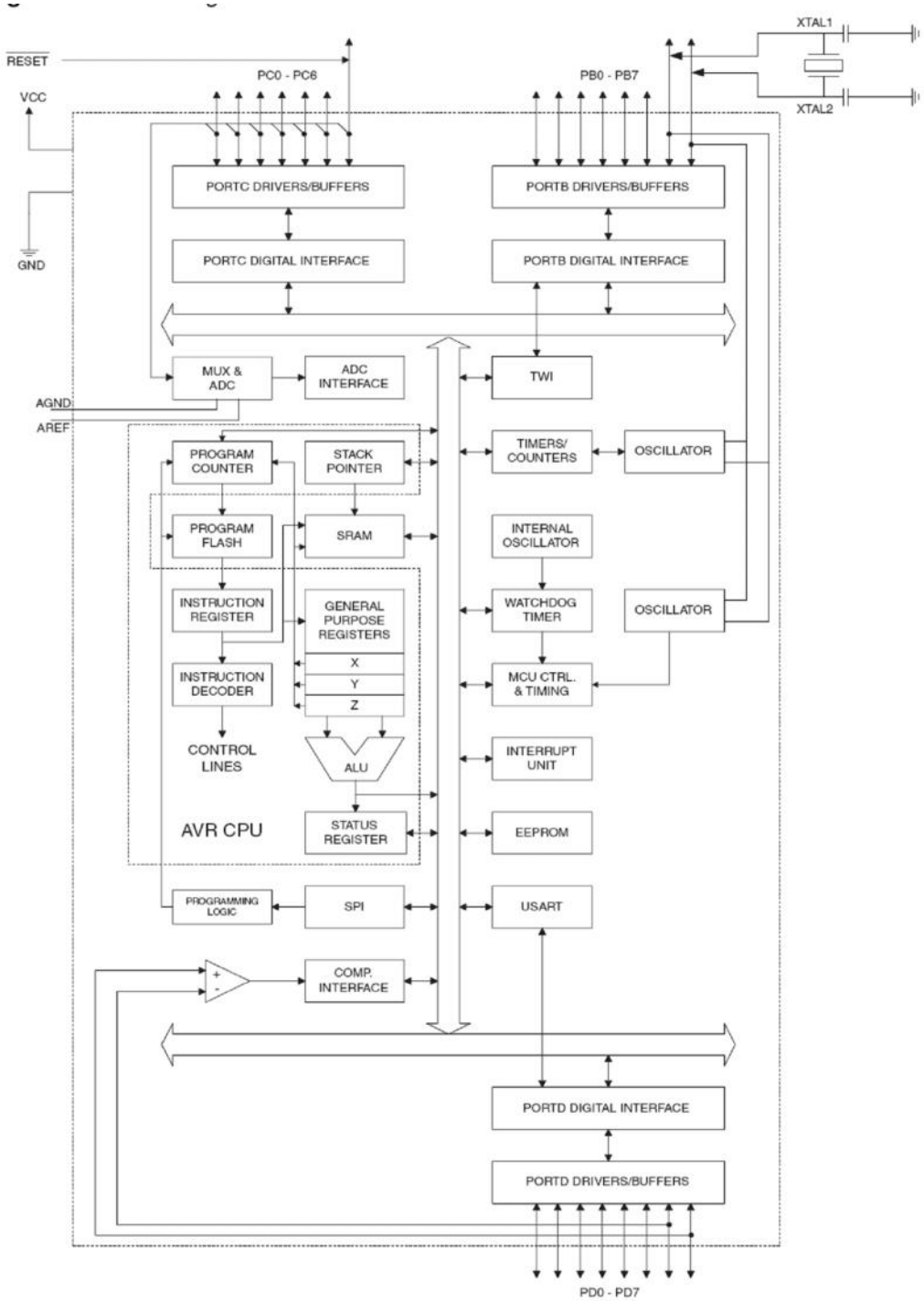


Обзор

ATmega8 – маломощный 8-разр. КМОП микроконтроллер, основанный на расширенной AVR RISC-архитектуре. За счет выполнения большинства инструкций за один машинный цикл ATmega8 достигает производительности 1 млн. операций в секунду/МГц, что позволяет проектировщикам систем оптимизировать соотношение энергопотребления и быстродействия.

Функциональная схема

Рис.1 Блок-схема



ATmega8 содержит следующие элементы: 8 кбайт внутрисистемно программируемой флэш-памяти с поддержкой чтения во время записи, 512 байт EEPROM, 1 кбайт статического ОЗУ, 23 линии универсального ввода-вывода, 32 универсальных рабочих регистра, счетчик реального времени (RTC), три гибких таймера-счетчика с режимами сравнения и ШИМ, 2 УСАПП, двухпроводной последовательный интерфейс ориентированный на передачу байт, 6-канальный(восемь каналов в TQFP и пакетах QFN/MLF) 10-разр. АЦП , программируемый сторожевой таймер с внутренним генератором, последовательный порт SPI, а также шесть программно выбираемых режимов уменьшения мощности. Режим холостого хода (Idle) останавливает ЦПУ, но при этом поддерживая работу статического ОЗУ, таймеров-счетчиков, SPI-порта и системы прерываний. Режим выключения (Powerdown) позволяет сохранить содержимое регистров, при остановленном генераторе и выключении встроенных функций до следующего прерывания или аппаратного сброса. В экономичном режиме (Power-save) асинхронный таймер продолжает работу, позволяя пользователю сохранить функцию счета времени в то время, когда остальная часть контроллера находится в состоянии сна. Режим снижения шумов АЦП (ADC Noise Reduction) останавливает ЦПУ и все модули ввода-вывода, кроме асинхронного таймера и АЦП для минимизации импульсных шумов в процессе преобразования АЦП. В дежурном режиме (Standby) кварцевый/резонаторный генератор продолжают работу, а остальная часть микроконтроллера находится в режиме сна. Данный режим характеризуется малой потребляемой мощностью, но при этом позволяет достичь самого быстрого возврата в рабочий режим.
Микроконтроллер производится по технологии высокоплотной энергонезависимой памяти компании Atmel. Встроенная внутрисистемно программируемая флэш-память позволяет перепрограммировать память программ непосредственно внутри системы через последовательный интерфейс SPI с помощью простого программатора или с помощью автономной программы в загрузочном секторе. Загрузочная программа может использовать любой интерфейс для загрузки прикладной программы во флэш-память. Программа в загрузочном секторе продолжает работу в процессе обновления прикладной секции флэш-памяти, тем самым поддерживая двухоперационность: чтение во время записи. За счет сочетания 8-разр. RISC ЦПУ с внутрисистемно самопрограммируемой флэш-памятью в одной микросхеме ATmega8 является мощным микроконтроллером, позволяющим достичь высокой степени гибкости и эффективной стоимости при проектировании большинства приложений встроенного управления. ATmega8 поддерживается полным набором программных и аппаратных средств для проектирования, в т.ч.: Си-компиляторы, макроассемблеры, программные отладчики/симуляторы, внутрисистемные эмуляторы и оценочные наборы.

Правовая оговорка

Типичные значения, содержавшиеся в этой спецификации, основаны на моделированиях и характеристике других микроконтроллеров AVR, произведенных на той же самой технологии . Мин. и Мак. значения будут доступны после того, как устройство будет характеризовано.

Описание выводов

VCC Напряжение питания цифровых элементов

GND Общий

Port B (PB7..PB0) XTAL1/XTAL2/TOSC1/TOSC2 Порт B – 8-разр. порт двунаправленного ввода-вывода с внутренними подтягивающими к плюсу резисторами (выбираются раздельно для каждого разряда). Выходные буферы порта B имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. При вводе, линии порта B будут действовать как источник тока, если внешне действует низкий уровень и включены подтягивающие резисторы. Выводы порта B находятся в третьем (высокоимпедансном) состоянии при выполнении условия сброса, даже если синхронизация не запущена**.**

В зависимости от настроек фьюзов генератора PB6 может быть как инверсный вход генератора и введен к внутренним часам через операционный усилитель. Если Внутренний Калиброванный генератор RC используется в качестве тактового источника микросхемы, PB7.. 6 используется в качестве TOSC2.. 1 ввод для Асинхронного Timer/Counter2, если бит AS2 в ASSR установлен.

Различные специальные функции Порта B рассматриваются в "Альтернативных Функциях Порта B" на странице 58 и "Системных Часах и Опциях Часов" на странице 25.

Port C (PC5..PC0) Порт C – 7-разр. порт двунаправленного ввода-вывода с внутренними подтягивающими к плюсу резисторами (выбираются раздельно для каждого разряда). Выходные буферы порта C имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. При вводе, линии порта C будут действовать как источник тока, если внешне действует низкий уровень и включены подтягивающие резисторы. Выводы порта C находятся в третьем (высокоимпедансном) состоянии при выполнении условия сброса, даже если синхронизация не запущена.

PC6/RESET Если фьюз RSTDISBL запрограммирован, PC6 используется в качестве контакта ввода-вывода. Отметьте, что электрические характеристики PC6 отличаются от таковых из других контактов Порта C. Если фьюз RSTDISBL непрограммируется, PC6 используется в качестве ввода Сброса. Низкий уровень на этом контакте дольше чем минимальная длина импульса генерирует Сброс, даже если генератор не будет работать. Минимальная длина импульса дается в Таблице 15 на странице 38. Более короткие импульсы, не гарантируют, генерацию Сброса. Различные специальные функции Порта C рассматриваются на странице 61.

Port D (PD7..PD0) Порт D – 8-разр. порт двунаправленного ввода-вывода с внутренними подтягивающими к плюсу резисторами (выбираются раздельно для каждого разряда). Выходные буферы порта D имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. При вводе, линии порта D будут действовать как источник тока, если внешне действует низкий уровень и включены подтягивающие резисторы. Выводы порта D находятся в третьем (высокоимпедансном) состоянии при выполнении условия сброса, даже если синхронизация не запущена.

Порт D также служит для различных специальных функций ATmega8 как перечислено на странице 63.

RESET Вход сброса. Если на этот вход приложить низкий уровень длительностью более минимально необходимой будет генерирован сброс независимо от работы синхронизации. Минимальная длительность внешнего импульса сброса приведена в таблице 15.

Действие импульса меньшей продолжительности не гарантирует генерацию сброса.

AVcc	вход питания аналогово-цифрового преобразователя Порт C (3.. 0), и ADC (7.. 6).. Он должен быть внешне связан с VCC, даже если АЦП не используется. При использовании АЦП этот вывод связан с VCC через фильтр низких частот. нижних частот. Отметьте что Порт C (5.. 4)может быть выходом цифрового напряжения VCC.
AREF	вход подключения источника опорного напряжения АЦП.

ADC7..6 (TQFP and QFN/MLF Package Only)	В TQFP и корпусе QFN/MLF, ADC7.. 6 служат аналоговыми входами к конвертеру A/D. Эти аналоговые контакты и служат 10-битовыми каналами ADC.

Ресурсы

Исчерпывающий набор средств разработки, указаний по применению и спецификаций доступен для загрузки на http://www.atmel.com/avr.

О примерах программ

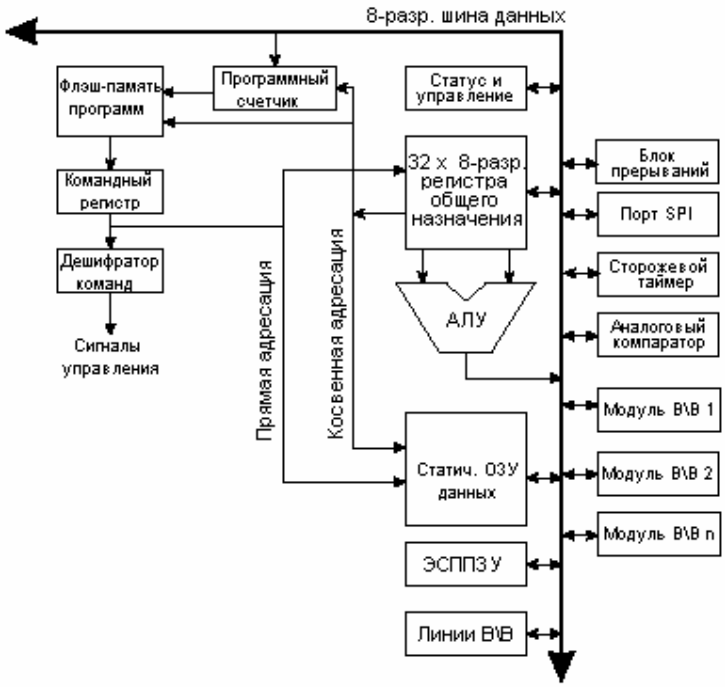
В данный документ входят примеры простых программ, которые кратко показывают как использовать различные составные части микроконтроллера. При составлении данных примеров предполагалось, что специфические файлы заголовков прописаны перед компиляцией. Следует понимать, что не все поставщики Си-компиляторов включают определения бит в файлы заголовков, а обработка прерываний в Си зависит от компилятора. Для уточнения этих особенностей см. документацию на используемый Си-компилятор.

Введение

В данном разделе описываются общие особенности архитектуры ядра AVR. Основная функция ядра ЦПУ заключается в гарантировании корректности выполнения программы. Помимо этого, ЦПУ должен иметь возможность адресоваться к различным видам памяти, выполнять вычисления, управлять периферийными устройствами и обрабатывать прерывания.

Краткий обзор архитектуры

Рис.2 Функциональная схема архитектуры AVR



Функциональная схема архитектуры AVR

В целях достижения максимальной производительности и параллелелизма у AVR-микроконтроллеров используется Гарвардская архитектура с раздельными памятью и шинами программ и данных. Команды в памяти программ выполняются с одноуровневой конвейеризацией. В процессе выполнения одной инструкции следующая предварительно считывается из памяти программ. Данная концепция позволяет выполнять одну инструкцию за один машинный цикл. Память программ представляет собой внутрисистемно программируемую флэш-память.
Регистровый файл с быстрым доступом содержит 32 x 8-разр. рабочих регистров общего назначения с однотактовым циклом доступа. Благодаря этому достигнута однотактность работы арифметико-логического устройства (АЛУ). При обычной работе АЛУ сначала из регистрового файла загружается два операнда, затем выполняется операция, а после результат отправляется обратно в регистровый файл и все это происходит за один машинный цикл.

6 регистров из 32 могут использоваться как три 16-разр. регистра косвенного адреса для эффективной адресации в пределах памяти данных. Один из этих указателей адреса может также использоваться как указатель адреса для доступа к таблице преобразования во флэш-памяти программ. Данные 16-разр. регистры называются X-регистр, Y-регистр и Z-регистр и описываются далее в этом разделе.

АЛУ поддерживает арифметические и логические операции между регистрами, а также между константой и регистром. Кроме того, АЛУ поддерживает действия с одним регистром. После выполнения арифметической операции регистр статуса обновляется для отображения результата выполнения операции.

Процесс выполнения программы обеспечивается условным и безусловным переходом и командами вызова, которые в состоянии непосредственно адресовать целое адресное пространство. У большинства инструкций AVR есть единственный 16-разрядный формат слова. Каждый адрес памяти Программы содержит 16-или 32-разрядную инструкцию.

Флэш-память программ разделена на две секции: секция программы начальной загрузки и секция прикладной программы. Обе секции имеют раздельные биты защиты от записи и чтения/записи. Инструкция SPM (запись в секцию прикладной программы) должна использоваться только внутри секции программы начальной загрузки.

При генерации прерывания и вызове подпрограмм адрес возврата из программного счетчика записывается в стек. Стек эффективно распределен в статическом ОЗУ памяти данных и, следовательно, размер стека ограничен общим размером статического ОЗУ и используемым его объемом. В любой программе сразу после сброса должна быть выполнена инициализация указателя стека (SP) (т.е. перед выполнением процедур обработки прерываний или вызовом подпрограмм). Указатель стека – SP – доступен на чтение и запись в пространстве ввода-вывода. Доступ к статическому ОЗУ данных может быть легко осуществлен через 5 различных режимов адресации архитектуры AVR.

Пространство памяти в архитектуре AVR - линейное и непрерывное.

Гибкий модуль прерываний содержит свои управляющие регистры в пространстве ввода-вывода и имеет дополнительный бит общего разрешения работы системы прерываний в регистре статуса. У всех прерываний имеется свой вектор прерывания в соответствии с таблицей векторов прерываний. Прерывания имеют приоритет в соответствии с позицией их вектора. Прерывания с меньшим адресом прерывания имеют более высокий приоритет.

Пространство памяти ввода-вывода содержит 64 адреса для периферийных функций ЦП как Регистры Управления, SPI, и другие функции ввода-вывода. К Памяти ввода -вывода можно получить доступ непосредственно, или как к памяти данных следующая за регистрами по адресам 0x20 - 0x5F.

АЛУ – арифметико-логическое устройство

Высокопроизводительное АЛУ AVR-микроконтроллеров работает в непосредственной связи со всеми 32 универсальными рабочими регистрами. АЛУ позволяет выполнить за один машинный цикл операцию между двумя регистрами или между регистром и константой. Операции АЛУ могут быть классифицированы на три группы: арифметические, логические и битовые. Кроме того, архитектурой ATmega8 поддерживаются операции умножения со знаком и без знака и дробным форматом. См. раздел "Набор инструкций" для подробного ознакомления.

Регистр статуса

Регистр статуса содержит информацию о результате только что выполненной арифметической инструкции. Данная информация может использоваться для ветвления программы по условию. Следует понимать, что регистр статуса обновляется после выполнения всех операций АЛУ в объеме предусмотренном для каждой конкретной инструкции (см. раздел Флаги в таблице инструкций). Флаги этого регистра в большинстве случаев позволяют отказаться от использования инструкций сравнения, делая код программы более компактным и быстрым. Обратите внимание, что состояние регистра статуса автоматически не запоминается при вызове процедуры обработки прерываний и не восстанавливается при выходе из нее. Это необходимо выполнить программно.

Регистр статуса SREG AVR-микроконтроллера имеет следующую структуру:

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Разряд 7 – I: Общее разрешение прерываний

Бит общего разрешения прерываний используется для активизации работы системы прерываний. Разрешение отдельных прерываний осуществляется в соответствующих управляющих регистрах. Если бит общего разрешения прерываний сбросить, то ни одно из прерываний не будет активным независимо от их индивидуальной конфигурации. Бит I сбрасывается в 0 аппаратно после генерации запроса на прерывание, а после выполнения инструкции возврата из прерывания RETI снова устанавливается к 1 для выполнения последующих прерываний. Бит I может также сбрасываться и устанавливаться с помощью инструкций CLI и SEI, соответственно.

Разряд 6 – T: Хранение копируемого бита

Специальные битовые операции BLD (копирование из T-бита) и BST (копирование в T-бит) используют в качестве источника и получателя данных бит T. Любой бит из регистрового файла может быть скопирован в бит T инструкцией BST, а также содержимое бита T может быть скопировано в любой бит регистрового файла с помощью инструкции BLD.

Разряд 5 – H: Флаг половинного переноса

Данный флаг устанавливается при выполнении некоторых арифметических инструкций и индицирует о возникновении половинного переноса. Как правило половинный перенос широко используется в двоично-десятичной арифметике. Более подробная информация приведена в описании набора инструкций.

Разряд 4 – S: бит знака, S = Искл. ИЛИ (N,?V)

Бит S – результат выполнения логической операции исключающего ИЛИ между флагом отрицательного результата N и флагом переполнения двоичного дополнения V. Более подробная информация приведена в описании набора инструкций.

Разряд 3 – V: Флаг переполнения двоичного дополнения

Флаг переполнения двоичного дополнения V поддерживает арифметику с двоичным дополнением. Более подробная информация приведена в описании набора инструкций.

Разряд 2 – N: Флаг отрицательного результата

Флаг отрицательного результата N индицирует, что результатом выполнения

арифметической или логической операции является отрицательное значение. Более подробная информация приведена в описании набора инструкций.

Разряд 1 – Z: Флаг нулевого результата

Флаг нулевого результата Z индицирует, что результатом выполнения арифметической или логической операции является ноль. Более подробная информация приведена в описании набора инструкций.

Разряд 0 – C: Флаг переноса

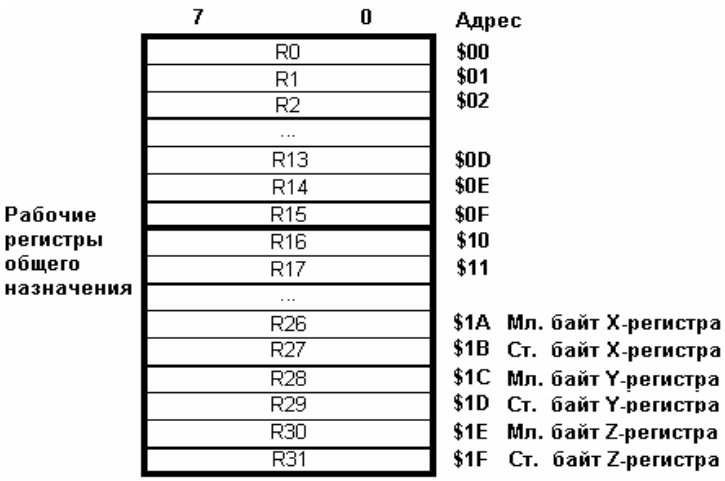
Флаг переноса C индицирует о возникновении переноса в результате выполнения арифметической или логической операции. Более подробная информация приведена в описании набора инструкций.

Файл регистров общего назначения

Файл регистров оптимизирован под расширенный набор инструкций AVR-микроконтроллеров. В целях достижения требуемой производительности и гибкости файлом регистров поддерживаются следующие схемы ввода-вывода:

- Один 8-разр. операнд и один 8-разр. результат
- Два 8-разр. операнда и один 8-разр. результат
- Два 8-разр. операнда и один 16-разр. результат
- Один 16-разр. операнд и один 16-разр. результат

Рисунок 3 показывает структуру 32 рабочих регистров общего назначения в ЦПУ.

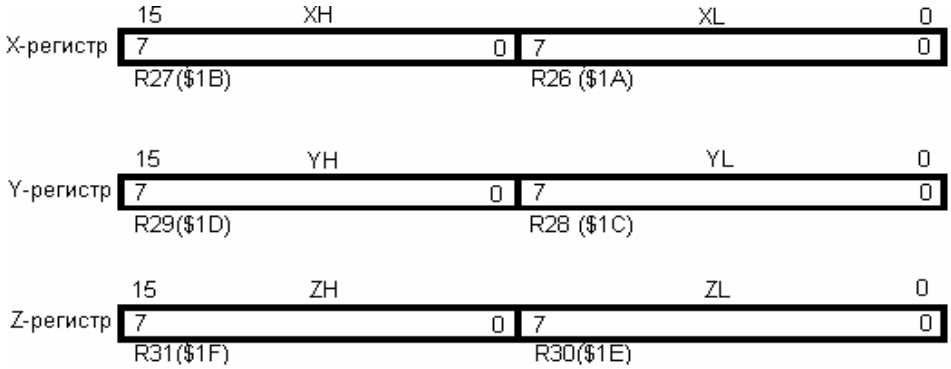


Большинство инструкций работающих с файлом регистров имеют непосредственный доступ ко всем регистрам, чем достигается выполнение их за один машинный цикл.

Как показано на рисунке 3, каждый регистр имеет свой адрес в области памяти данных, для чего отведено там первые 32 позиции. Не смотря на физическую реализацию не по адресам статического ОЗУ, данная архитектура памяти обеспечивает высокую гибкость доступа к регистрам, например, регистры-указатели X, Y и Z могут быть установлены для присвоения индекса любому регистру в файле.

X-регистр, Y-регистр и Z-регистр

Регистры R26..R31 обладают некоторым дополнительными функциями для их общецелевого использования. Данные регистры являются 16-разр. указателями адреса для косвенной адресации в пределах памяти данных
Три регистра косвенной адресации X, Y и Z представлены на **рисунке 4**



В различных режимах адресации данные адресные регистры выполняют функции фиксированного смещения, автоматического инкрементирования и автоматического декрементирования (см. описание набора инструкций для более подробного изучения).

Указатель стека

Указатель стека указывает на область стека в статическом ОЗУ данных, где размещены стеки прерываний и подпрограммы. Данная область стека в статическом ОЗУ памяти данных должна быть определена программно до вызова любой процедуры или разрешения прерываний. Устанавливаемое значение указателя стека должно быть более $60. Указатель стека однократно декрементируется при помещении данных в стек инструкцией PUSH и дважды декрементируется при помещении в стек адреса возврата при вызове подпрограмм или прерываниях. Указатель стека однократно инкрементируется при извлечении данных из стека инструкцией POP и дважды инкрементируется при извлечении адреса возврата при выполнении инструкции выхода из подпрограммы RET или выхода из процедуры обработки прерываний RETI. Указатель стека реализован как два 8-разр. регистра в области ввода-вывода. Число фактически используемых разрядов зависит от типа микроконтроллера. Обратите внимание, что у некоторых AVR-микроконтроллеров область памяти данных настолько мала, что достаточно только регистра SPL. В этом случае регистр SPH отсутствует.

Разряды	15	14	13	12	11	10	9	8	
	SP15	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
	SP7	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
	7	6	5	4	3	2	1	0	
Чтение/Запись	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	
	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	
Начальное значение	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Временная диаграмма выполнения инструкции

ЦПУ AVR-микроконтроллера тактируется сигналом CLKЦПУ, который непосредственно генерируется выбранным источником синхронизации. Внутреннее деление тактовой частоты не используется.

Рисунок 5 показывает параллельность выборок и исполнения инструкций, что обеспечивается Гарвардской архитектурой и концепцией регистрового файла с быстрым доступом. Данная концепция конвейеризации обеспечивает удельную производительности 1 млн.оп в сек./МГц и предоставляет уникальное соотношение числа функций на стоимость, число функций на такт синхронизации и числа функций на Вт потребляемой мощности.

Рисунок 5 – Параллельные выборки и исполнения инструкций

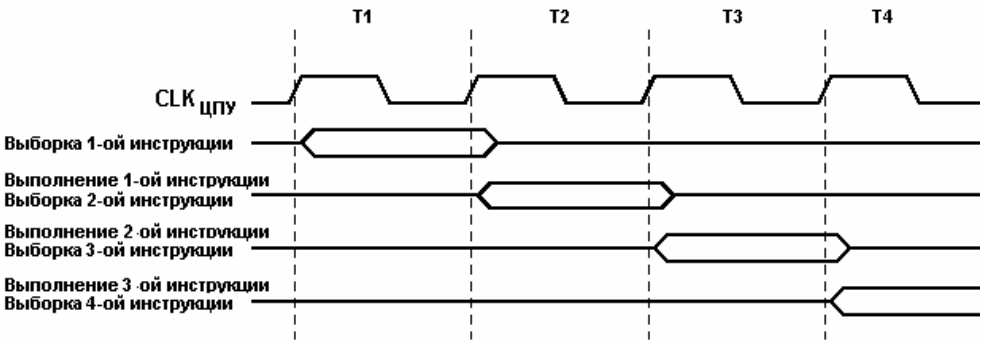
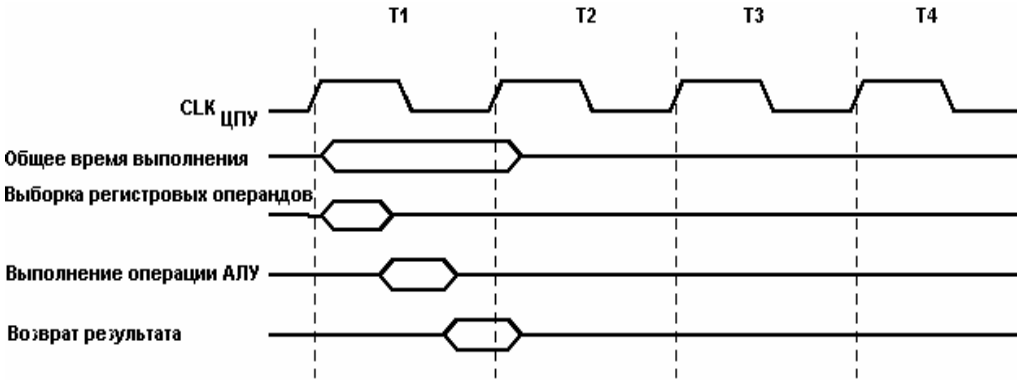


Рисунок 6 иллюстрирует концепцию внутренней временной диаграммы для регистрового файла. За один такт синхронизации АЛУ выполняет действие над двухрегистровым операндом и возвращает результат обратно в регистр-получатель.



Сброс и обработка прерываний

AVR-микроконтроллеры поддерживают несколько различных источников прерываний. Все прерывания, а также сброс имеют свой индивидуальный вектор в памяти программ. Для каждого прерывания имеется собственный бит разрешения. Кроме того, имеется возможность общего разрешения работы прерываний с помощью управления соответствующим битом в статусном регистре. В зависимости от значения программного счетчика прерывания могут быть автоматически отключены, если запрограммировать биты защиты загрузочного сектора BLB02 или BLB12. Данная функция улучшает защиту программы. См. раздел “Программирование памяти” для уточнения деталей.

Наименьшие адреса в памяти программ по умолчанию определены как вектора сброса и прерываний. Полный перечень векторов приведен в разделе "Прерывания". В перечне также определяется уровень приоритетов различных прерываний. Меньшие адреса обладают более высоким уровнем приоритетом. Сброс (RESET) имеет наивысший приоритет, за ним следует INT0 – запрос на внешнее прерывание по входу INT0. Векторы прерывания могут быть перемещены в начало загрузочного сектора флэш-памяти установкой бита IVSEL в регистре управления микроконтроллером (MCUCR). См. раздел "Прерывания” для более подробного ознакомления. Вектор сброса может быть также перемещен в начало загрузочного сектора флэш-памяти путем программирования конфигурационного бита BOOTRST (см. “Самопрограммирование из сектора начальной загрузки с поддержкой чтения во время записи”).

После возникновения прерывания бит I общего разрешения прерываний сбрасывается и все прерывания запрещаются. Пользователь может программно записать лог. 1 в бит I для разрешения вложенных прерываний. В этом случае все разрешенные прерывания могут прервать текущую процедуру обработки прерываний. Бит I автоматически устанавливается после выполнения инструкции выхода из прерывания RETI.

Имеется два основных типа прерываний. Первый тип прерываний активизируется событием, которое приводит к установке флага прерываний. Для данных прерываний программный счетчик изменяется на соответствующий вектор прерывания для выполнения процедуры его обработки и затем аппаратно очищает флаг прерывания. Флаги прерывания также сбрасываются путем записи лог.1 в соответствующий разряд. Если возникает условие прерывания, но данное прерывание запрещено, то флаг устанавливается и запоминается до разрешения этого прерывания или сбрасывается программно. Аналогично, если возникает одно и более условий прерываний при сброшенном флаге общего разрешения прерываний, то соответствующий флаг устанавливается и запоминается до возобновления работы прерываний, а затем прерывания будут выполнены в соответствии с приоритетом.

Второй тип прерываний активизируется сразу после выполнения условия прерывания. Данные прерывания не обязательно имеют флаги прерываний. Если условие прерывания исчезает до его разрешения, то данный запрос игнорируется. После выхода из прерывания AVR-микроконтроллер возвращается к выполнению основной программы и выполняет еще одну инструкцию до обслуживания любого из отложенных прерываний.

Обратите внимание, что регистр статуса автоматически не запоминается при вызове процедуры обработки прерывания и не восстанавливается при выходе из этой процедуры. Данные действия необходимо выполнить программно.

При выполнении инструкции CLI все прерывания запрещаются. Запрос на прерывание не будет отработан после выполнения инструкции CLI, даже если оно возникает одновременно с выполнением команды CLI. В следующем примере показано как избежать прерываний во время выполнения временной последовательности записи в EEPROM.

Пример кода на Ассемблере

```
in r16, SREG ; Запомнили состояние регистра статуса SREG
cli ; отключаем все прерывания во время отработки временной
последовательности
sbi EECR, EEMWE ; Разрешаем запись в EEPROM
sbi EECR, EEWE
out SREG, r16 ; Восстанавливаем значение SREG (бит I)
```

Пример кода на Си

```
char cSREG;
cSREG = SREG; /* Запоминаем значение SREG */
/* Отключение прерываний на время задания временной последовательности
*/
 _CLI();
EECR |= (1<<EEMWE); /* Старт записи в  EEPROM */
EECR |= (1<<EEWE);
SREG = cSREG; /* Восстанавливаем значение SREG (бит I) */
```

Для разрешения прерываний используется инструкция SEI, а следующая за SEI инструкция будет выполнена перед отработкой любого отложенного прерывания, как показано в примере.

Пример кода на Ассемблере

```
sei ; Общее разрешение прерываний
sleep ; перевод в режим ожидания прерывания
;Прим.: Режим ожидания будет введен прежде чем запустится отработка
 ; отложенного   прерывания
```

Пример кода на Си

```
_SEI(); /* Общее разрешение прерываний */
_SLEEP(); /* перевод в режим ожидания прерывания */
/*Прим.:Режим ожидания будет введен прежде чем запустится отработка
/* отложенного прерывания */
```

Время реакции на прерывание Реакция на отработку запроса на прерывание длится минимум 4 машинных цикла. По истечении этого времени программа продолжает свое выполнение с вектора соответствующего прерывания. В течение 4 машинных циклов состояние программного счетчика помещается в стек. Как правило, по адресу вектора прерываний хранится команда перехода на процедуру обработку прерываний, а на данный переход затрачивается еще 3 машинных цикла. Если запрос на прерывание возникает в процессе исполнения инструкции, требующей более 1 машинного цикла на выполнение, то прерывание будет обработано только после выполнения этой инструкции. Если прерывание возникает во время нахождения микроконтроллера в режиме сна, то реакция на прерывание увеличится еще на 4 цикла. Данная задержка связана с временем старта из выбранного режима сна.

Выход из процедуры обработки прерывания требует 4 машинных цикла. В течение этого времени двухбайтный программный счетчик извлекается из стека, указатель стека дважды инкрементируется и устанавливается бит I в регистре статуса SREG.

Память ATMEGA8

В данном разделе описываются различные виды памяти ATmega8. Память AVR-микроконтроллера разделена на две области: память данных и память программ. Кроме того, ATmega8 содержит память на EEPROM для энергонезависимого хранения данных. Все три области памяти являются линейными и равномерными.

Внутрисистемно Программируемая флэш-память программ

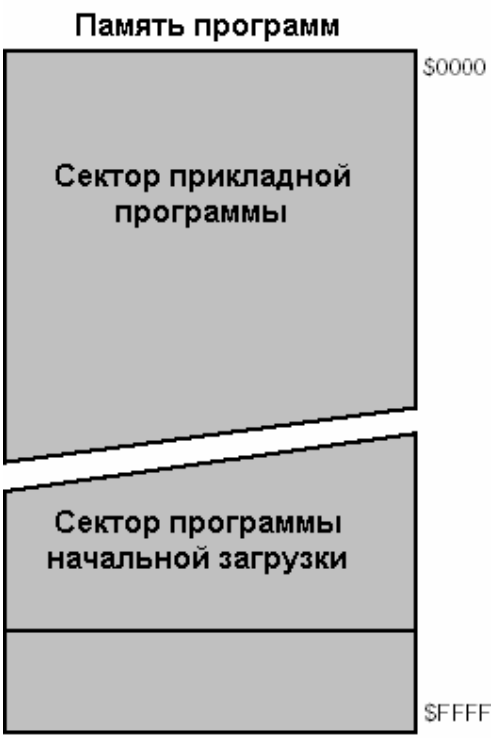
ATmega8 содержит 8 кбайт внутренней внутрисистемно перепрограммируемой флэш-памяти для хранения программы. Поскольку все AVR-инструкции являются 16 или 32-разр., то флэш-память организована как 4 кбит x 16. Для программной защиты флэш-память программ разделена на два сектора: сектор программы начальной загрузки и сектор прикладной программы**.**

Флэш-память характеризуется износостойкостью не менее 10000 циклов запись/стирание. Программный счетчик PC у ATmega8 является 12-разр., поэтому, позволяет адресоваться к 4 кбайт памяти программ. Работа сектора программы начальной загрузки и связанных с ним бит защиты программы детально описана в разделе “Самопрограммирование из сектора начальной загрузки с поддержкой чтения во время записи”стр209. В разделе “Программирование памяти”стр222 детально описывается параллельное программирование флэш-памяти и последовательное программирование через интерфейсы SPI, JTAG.

Таблицы констант могут располагаться в пределах всего пространства памяти программ (см. описание инструкции чтения из памяти программ LPM и расширенного чтения из памяти программ ELPM).

Временные диаграммы выборки и исполнения инструкций представлены в разделе “Временная диаграмма выполнения инструкции”. Стр14.

Рис.7



Статическое ОЗУ памяти данных

Рисунок 8 показывает, как ATmega8 организуется SRAM Память.
Первые 1020 ячеек памяти данных относятся к файлу регистров, памяти ввода-вывода и внутреннему статическому ОЗУ данных. В первых 96 ячейках расположен файл регистров и стандартная память ввода-вывода и следующие 1024 ячейки занимает внутреннее ОЗУ данных.
Реализовано пять различных способов адресации для охвата всей памяти: прямая, косвенная со смещением, косвенная, косвенная с предварительным декрементом и косвенная с последующим инкрементом. Регистры R26…R31 из файла регистров используются как регистры-указатели для косвенной адресации.

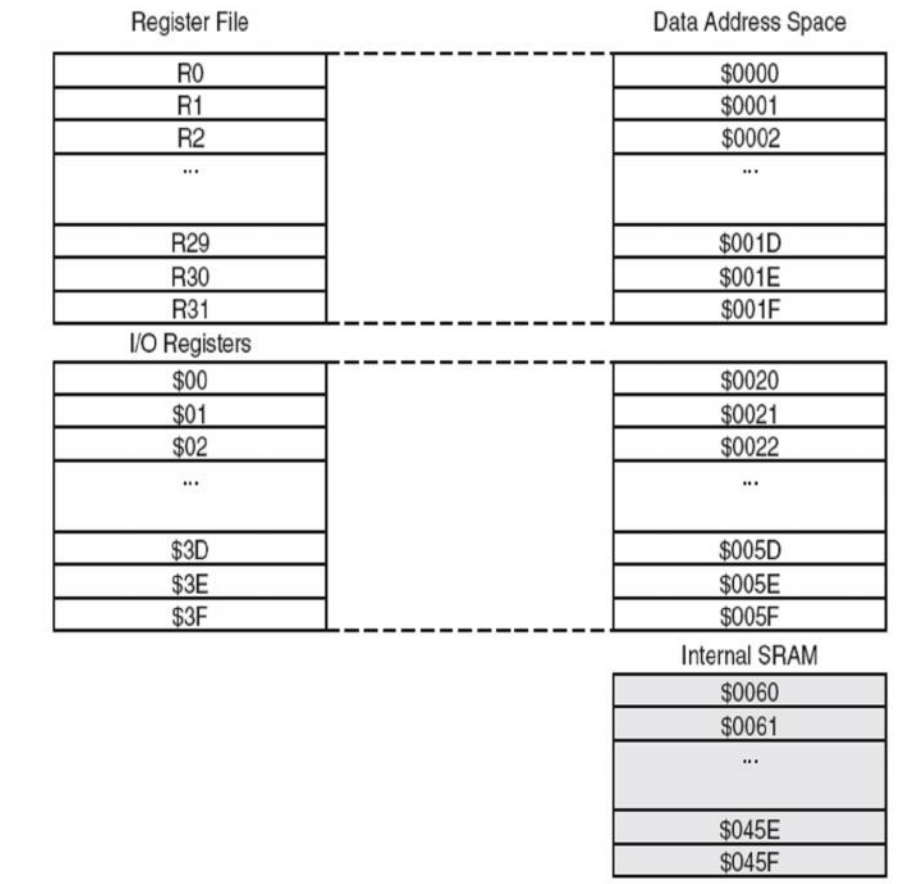
Прямая адресация позволяет адресоваться ко всей памяти данных.

Косвенная адресация со смещением позволяет адресовать 63 ячейки, начиная с адреса указанного в регистрах Y или Z.

При использовании инструкции косвенной адресации с предварительным декрементом и последующим инкрементом значения адресных регистров X, Y и Z, соответственно декрементируются до или инкрементируются после выполнения инструкции.

32 рабочих регистров общего назначения, 64 регистра ввода-вывода и 1024 байт внутреннего статического ОЗУ данных в ATmega8 доступны с помощью всех этих режимов адресации. Файл регистров описывается в разделе “Файл регистров общего назначения”стр12.

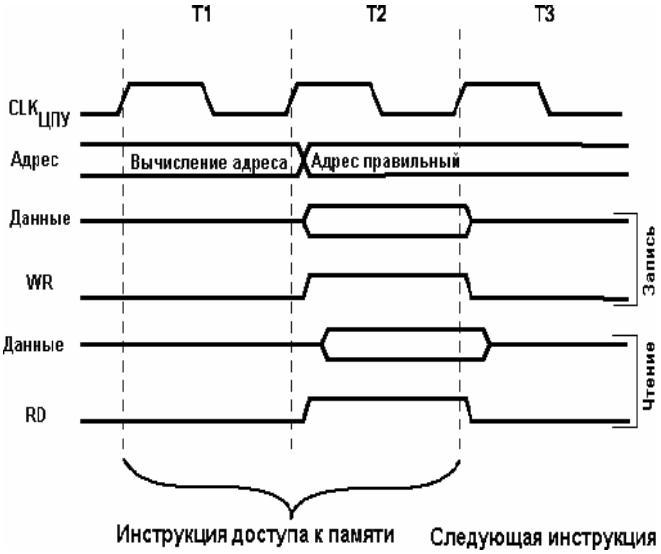
Рис.8 Карта памяти данных



Временная диаграмма доступа к памяти

В данном разделе описывается общая концепция доступа к внутренней памяти.

Доступ к внутреннему статическому ОЗУ выполняется за два машинных цикла в соответствии с **рисунком 9**.



Память данных на EEPROM

ATmega8 содержит 512 байт памяти данных на EEPROM,. Она организована как отдельная область памяти данных, в которой один байт может быть записан и считан. EEPROM, характеризуется износостойкостью 100000 циклов чтения/записи. Доступ к EEPROM, осуществляется через специальные регистр адреса EEPROM,,регистр данных EEPROM, и регистр управления.

“Программирование памяти” на странице 222 содержит подробное описание на EEPROM, Программирование в SPI - или режим Параллельного программирования.

Доступ Чтения-записи EEPROM

Время записи в EEPROM приведено в табл. 2. Функция самосинхронизации позволяет программно определить возможность записи следующего байта. Если код программы содержит инструкции записи в EEPROM, то должны быть приняты следующие меры предосторожности. У источников питания с хорошей фильтрацией напряжение VCC медленно нарастает/спадает при подаче/снятии питания. По этой причине микроконтроллер в течение некоторого периода времени может оказаться под меньшим напряжением питания, чем требуется для заданной тактовой частоты. См. раздел “Предотвращение повреждения данных в EEPROM ” для детального изучения методов разрешения данной проблемы.стр.23

В целях предотвращения неумышленной записи в EEPROM должна быть выполнена специфическая процедура записи. Детально этот вопрос рассматривается при описании Управляющего регистра EEPROM.

Когда происходит считывание EEPROM ЦПУ задерживается на 4 машинных цикла до выполнения следующей инструкции. Во время записи в EEPROM ЦПУ задерживается на два машинных цикла до выполнения следующей инструкции.

Адресные регистры EEPROM – EEARH и EEARL

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	–	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

Разряды 15..9– Резерв

Данные зарезервированные разряды считываются как 0.

Разряды 8..0 – EEAR11..0: Адрес ячейки EEPROM

Регистры адреса EEPROM – EEARH и EEARL – определяют адрес ячейки EEPROM в 512 байтном пространстве. Байтные ячейки EEPROM адресуются линейно в диапазоне адресов 0…511. Начальное значение EEAR неопределенное. Необходимое значение адреса должно быть записано до начала доступа к EEPROM.

Регистр данных EEPROM – EEDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Разряды 7…0 – EEDR7.0: Данные EEPROM

Для выполнения записи в EEPROM в регистр EEDR необходимо указать записываемые данные, которые будут записаны по адресу, указанному в регистре EEAR. После выполнения чтения из EEPROM в регистре EEDR содержатся считанные данные из ячейки по адресу указанному в EEAR.

Регистр управления EEPROM – EECR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	X	0	

Разряды 7…4 – Резерв

Данные разряды у ATmega8 зарезервированы и считываются как 0.

Разряд 3 – **EERIE**: Разрешение прерывания по готовности EEPROM

Запись в EERIE 1 разрешает прерывание по готовности EEPROM, если кроме того установлен бит I в регистре SREG. Запись в EERIE нуля отключает это прерывание. Прерывание по готовности EEPROM генерируется, если бит EEWE сброшен.

Разряд 2 – **EEMWE**: Главное разрешение записи в EEPROM

Бит EEMWE разрешает установку бита EEWE, инициирующего запись в EEPROM. Данные будут записаны в EEPROM по указанному адресу, если в EEMWE записать 1, а затем в течение 4 машинных циклов записать 1 в EEWE. Если EEMWE=0, то запись в EEWE лог. 1 не вызовет никаких действий. После программной установки бита EEMWE он автоматически сбрасывается аппаратно по истечении четырех машинных циклов.

Разряд 1 – **EEWE**: Разрешение записи в EEPROM

Сигнал разрешения записи EEWE является стробирующим сигналом записи для EEPROM.

Для записи в EEPROM после корректной установки адреса и данных необходимо установить бит EEWE.

Перед установкой бита EEWE должен быть установлен бит EEWE, иначе запись в EEPROM не произойдет.
При выполнении операции записи в EEPROM необходимо руководствоваться следующей последовательностью (порядок шагов 3 и 4 не важен):

1.Ожидание пока EEWE станет равным нулю.
2.Ожидание равенства нулю бита SPMEN в регистре SPMCSR.
3.Запись нового адреса ЭСППЗУ в EEAR (опционально).
4.Запись новых данных в регистр EEDR для записи в EEPROM (опционально).
5.Запись лог. 1 в EEMWE, когда в EEWE регистра EECR записан ноль.
6.Запись лог. 1 в EEWE в течение четырех машинных циклов после установки EEMWE.

EEPROM нельзя программировать во время записи флэш-памяти из ЦПУ. С учетом этого, перед началом новой записи в EEPROM необходимо проверить завершение программирования флэш-памяти. Шаг 2 необходимо выполнять, если в приложении используется программирование из загрузочного сектора. Если программирование флэш-памяти под управлением ЦПУ не предусмотрено, то шаг 2 может быть исключен. См. “Самопрограммирование из сектора начальной загрузки с поддержкой чтения во время записи” для детального изучения программирования из загрузочного сектора.стр.209

Предостережения: Прерывание между шагами 5 и 6 может нарушить цикл записи из-за превышения установленного предела времени на выполнение этих шагов. Если процедура обработки прерывания, осуществляющая доступ к EEPROM, прерывается другим доступом к EEPROM, то EEAR или EEDR будут изменены, вызывая сбой прерванного цикла доступа. Во избежание этих проблем рекомендуется сбрасывать флаг общего разрешения прерываний при выполнении последних четырех шагов.

По окончании записи бит EEWE сбрасывается аппаратно. Данный бит может опрашиваться программно для определения возможности записи следующего байта (нулевое значение). После установки EEWE ЦПУ останавливается на два машинных цикла перед выполнением следующей инструкции.

Разряд 0 – EERE: Разрешение чтения из EEPROM

Сигнал разрешения чтения из EEPROM EERE является стробом чтения EEPROM. После записи корректного адреса в регистр адреса EEAR бит EERE должен быть установлен к лог.1 для запуска механизма чтения EEPROM. Чтение из EEPROM выполняется одновременно с инструкцией, поэтому, запрашиваемые данные готовы для считывания сразу по ее завершении. После чтения из EEPROM ЦПУ задерживается на четыре машинных цикла, а только затем выполняет следующую инструкцию.

Пользователь должен опросить флаг EEWE до начала операции чтения. Если осуществляется операция записи, то невозможно не только считать EEPROM, но и изменить регистр адреса EEAR. Во время доступа к EEPROM используется калиброванный генератор. В таблице 1 приведено типичное время программирования EEPROM через ЦПУ.

Таблица 1 – Время программирования EEPROM

Операция программирования	Количество периодов калиброванного RC-генератора (1).	Типичное время
Запись EEPROM (через ЦПУ)	8448	8.5 мс

Прим 1. Используется частота 1 МГц независимо от установки конфигурационного бита CKSEL

Далее представлены примеры кодов функций записи в EEPROM на языках Ассемблер и Си. В данных примерах предполагается, что прерывания работают таким образом, что ни одно не возникает в процессе выполнения данных функций (например, путем общего отключения прерываний). Кроме того, считается, что из загрузочного сектора не выполняется программирование флэш-памяти. В противном случае функция записи в EEPROM должна ожидать окончания действия инструкции SPM.

Пример кода на Ассемблере

```
EEPROM_write:
; Ожидаем окончание предыдущей записи
sbic EECR,EEWE
rjmp EEPROM_write
; Записываем адрес (r18:r17) в адресный регистр EEPROM
out EEARH, r18
out EEARL, r17
; Записываем данные (r16)  в регистр данных EEPROM
out EEDR,r16
; Записывает лог. 1 в EEMWE
sbi EECR,EEMWE
; Запуск записи в EEPROM установкой EEWE
 sbi EECR,EEWE
   ret
```

Пример кода на Си

```
 void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
   {
/* Ожидаем окончание предыдущей записи */
while(EECR & (1<<EEWE))
 ;
 /* Указание адреса и данных */
  EEAR = uiAddress;
  EEDR = ucData;
 /* Запись лог. 1 в  EEMWE */
 EECR |= (1<<EEMWE);
  /* Запуск записи в EEPROM путем установки EEWE */
EECR |= (1<<EEWE);
 }
```

В следующих примерах кодов на Си и Ассемблере представлены функции чтения из EEPROM. При разработке примеров учитывалось управление прерываниями таким образом, что ни одно из них не возникает в процессе выполнения этих функций.

Пример кода на Ассемблере

```
EEPROM_read:
; Ожидание завершения предыдущей записи
 sbic EECR,EEWE
 rjmp EEPROM_read
 ; Установка адреса (r18:r17) в адресном регистре
out EEARH, r18
out EEARL, r17
;Запуск чтения EEPROM путем установки EERE
 sbi EECR,EERE
 ; Считывание данных из регистра данных EEPROM
 in r16,EEDR
 ret
```

Пример кода на Си

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
 /* Ожидание  завершения предыдущей записи*/
while(EECR & (1<<EEWE))
 ;
 /* Установка адресного регистра */
EEAR = uiAddress;
/* Разрешение чтения из EEPROM путем установки EERE */
EECR |= (1<<EERE);
/* Возврат данных из регистра данных EEPROM */
 return EEDR;
   }
```

Запись в EEPROM в режиме выключения (Power Down)

Если микроконтроллер переводится в режим выключения (Power Down) командой sleep в процессе выполнения операции записи в EEPROM, то операция записи будет продолжена и завершится по истечении требуемого времени доступа для записи. Однако, по завершении операции записи кварцевый генератор будет продолжать работу, и как следствие, микроконтроллер будет переведен в режим снижения мощности не полностью. С учетом этого, рекомендуется проверять окончание операции записи в EEPROM перед переводом в режим выключения (Power-down).

Меры предотвращения повреждения данных в EEPROM

В те моменты, когда напряжение VCC находится на уровне недостаточном для корректной работы ЦПУ и EEPROM, содержимое EEPROM может быть нарушено. Данные проблемы аналогичны устройствам, использующих отдельное EEPROM, поэтому, в данном случае необходимо применить те же меры.

При сниженном напряжении питания может быть две причины нарушения содержимого EEPROM. Первая причина состоит в возможности корректной регулярной записи в EEPROM только при определенном напряжении питания. Вторая состоит в возможности некорректного выполнения программы микроконтроллером при чрезмерном низком уровне питания.

Повреждение данных в EEPROM может быть легко предотвращено, если придерживаться следующих рекомендаций:

Микроконтроллер необходимо удерживать в состоянии сброса (низкий уровень на выводе RESET) при недостаточности уровня питания. Аналогично это можно выполнить, разрешив работу встроенного детектора питания (BOD). Если пороговый уровень встроенного детектора питания не соответствует необходимому порогу, то следует применить внешнюю схему сброса при снижении VCC (супервизор питания). Если сброс возникает во время действия операции записи, то запись будет завершена при условии достаточности уровня питания.

Память ввода-вывода

Существующее пространство ввода-вывода для ATmega8 показано в разделе “Сводная таблица регистров”.

Все порты ввода-вывода и периферийные устройства в ATmega8 размещены в пространстве ввода-вывода. Доступ ко всем ячейкам ввода-вывода может быть осуществлен с помощью инструкций IN и OUT путем передачи данных между одним из 32-х универсальным рабочим регистром и памятью ввода-вывода. Регистры ввода-вывода с адресами $00 - $1F могут побитно адресоваться с помощью инструкций SBI и CBI. Состояние одного из разрядов в этих регистрах может тестироваться с помощью инструкций SBIS и SBIC. При использовании специфических команд ввода-вывода IN и OUT необходимо использовать адреса $00 - $3F. Если адресоваться к регистрам ввода-вывода как к памяти данных с помощью инструкций LD и ST, то к указанным выше адресам необходимо прибавить $20.

Если осуществляется доступ к зарезервированным разрядам, то в целях совместимости с последующими микроконтроллерами в них необходимо записывать лог. 0.

Не должна производиться запись в ячейки по зарезервированным адресам в пространстве ввода-вывода.

Некоторые флаги статуса сбрасываются путем записи в них лог. 1. Обратите внимание, что инструкции CBI и SBI работают со всеми разрядами регистра ввода-вывода (чтение-модификация-запись).

Инструкции CBI и SBI работают только с регистрами по адресам $00…$1F.

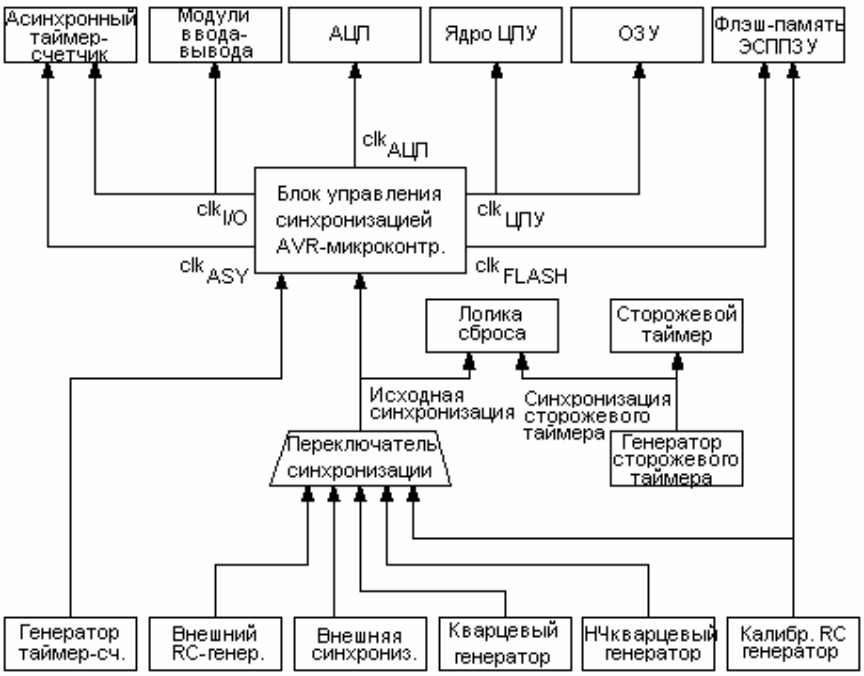
Регистры управления вводом-выводом и периферийными устройствами описываются в следующих разделах.

Системная синхронизация и тактовые источники

Источники синхронизации и их распределение

На рисунке 18 представлены источники синхронизации и распределение синхроимпульсов к встроенным блокам ATmega8. Не обязательно вся синхронизация должна работать в одно и тоже время. В целях снижения энергопотребления тактирование неиспользуемых модулей может быть прекращено путем перевода в различные режимы сна командой sleep (см. “Управление энергопотреблением и режимы сна”).

Рисунок 10 – Распределение синхронизирующих импульсов



Синхронизация ЦПУ – clkCPU

Синхронизация ЦПУ подключается к модулям микроконтроллера, которые связаны с работой ядра AVR. Примерами таких модулей являются файл регистров общего назначения, регистр статуса и память данных, выполняющая функцию стека. Остановка синхронизации ЦПУ приводит к прекращению выполнения ядром любых действий и вычислений.

Синхронизация ввода-вывода – clkI/O

Синхронизация ввода-вывода используется основными модулями ввода-вывода, в т.ч. таймеры-счетчики, SPI и УСАПП. Она также используется модулем внешних прерываний, но в некоторых случаях внешние прерывания детектируются в асинхронном режиме для поддержки работоспособности внешних прерываний даже при отключенной синхронизации. Также обратите внимание, что после отключения данной синхронизации (во всех режимах сна) двухпроводной интерфейс TWI продолжает наблюдать за передаваемым по шине адресом асинхронно.

Синхронизация флэш-памяти – clkFLASH

Синхронизация флэш-памяти тактирует работу интерфейса флэш-памяти. Обычно эта синхронизация работает одновременно с синхронизацией ЦПУ.

Синхронизация асинхронного таймера – clkASY

Синхронизация асинхронного таймера используется для тактирования асинхронного таймера-счетчика внешним кварцевым резонатором частотой 32 кГц. Данный тактовый генератор позволяет использовать таймер-счетчик как счетчик реального времени, даже при переводе микроконтроллера в режим сна. Асинхронный Timer/Counter использует те же штырьки XTAL как основу синхронизиронизации CPU но требует основной частоты часов CPU более, чем четыре раза частоту Генератора. Таким образом, асинхронная работа доступна только пока чип синхронизирован на Внутреннем Генераторе.

Синхронизация АЦП – clkADC

АЦП тактируется обособленным блоком синхронизации. Это позволяет остановить работу синхронизации ЦПУ и ввода-вывода на время преобразования АЦП в целях снижения влияния цифрового шума на результат преобразования. С помощью этого достигается более точный результат преобразования.

Источники синхронизации

С помощью конфигурационных бит имеется возможность выбора нескольких источников синхронизации. Сигнал синхронизации выбранного источника является входным для тактового генератора AVR и затем подключается к соответствующим модулям.

Таблица 2 – Выбор опций синхронизации микроконтроллера

Источники синхронизации	CKSEL3..0(1)
Внешний кварцевый/керамический резонатор	1111 – 1010
Внешний низкочастотный кварцевый резонатор	1001
Внешний RC-генератор 1000 – 0101	
Встроенный калиброванный RC-генератор	0100 – 0001
Внешняя синхронизация	0000

Прим. 1: Для всех конфигурационных бит “1” означает незапрограммированное состояние, а “0” – запрограммированное.

Подробное описание каждой из этих опций приведено в следующих разделах. При выходе ЦПУ из режима выключения (Power-down) или экономичного режима (Power-save) выбранный источник синхронизации используется по истечении времени на запуск, тем самым гарантируя стабильность работы генератора перед первым выполнением инструкции. Запуск микроконтроллера, инициированный сбросом (reset), сопровождается дополнительной задержкой для достижения питанием стабильного уровня перед переводом микроконтроллера в нормальный режим работы. Генератор сторожевого таймера используется для синхронизации данного модуля, который формирует задержку при запуске. Длительность генерируемой задержки определяется количеством импульсов генератора сторожевого таймера и для различных случаев приведена в таблице 3. Частота генератора сторожевого таймера зависит от напряжения питания, что показано в разделе “Типовые характеристики ATmega8: предварительные данные”. Устройство поставляется с CKSEL = “0001” и SUT = “10” (Внутренний генератор RC на 1 МГц, медленно возрастающее питание).

Таблица 3 – Количество тактов сторожевого таймера

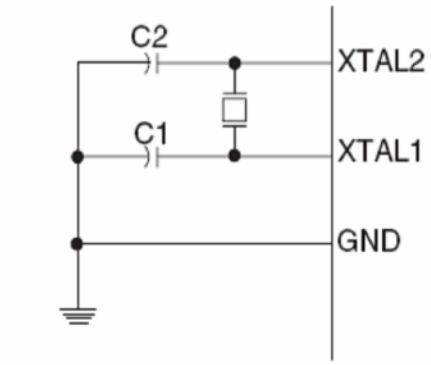
Типичное время переполнения (VCC = 5.0В)	Типичное время переполнения (VCC = 3.0В)	Количество тактов
4.1 мс	4.3 мс	4K (4096)
65 мс	69 мс	64K (65536)

Кварцевый генератор

XTAL1 и XTAL2 – вход и выход, соответственно, инвертирующего усилителя, который может быть настроен для использования в качестве встроенного генератора (см. рисунок 11). Для задания частоты может использоваться либо кварцевый либо керамический резонатор. Конфигурационный бит CKOPT выбирает один из двух режимов усилителя генератора. Если CKOPT запрограммирован, то амплитуда колебаний выходного сигнала генератора будет ограничена уровнями питания. Данный режим рекомендуется использовать при высоком уровне окружающих шумов или при использовании выхода XTAL2 в качестве источника синхронизации внешней схемы. Данный режим характеризуется широким частотным диапазоном. Если CKOPT – незапрограммирован, то амплитуда выходных колебаний генератора снижается. Использование данного режима позволяет существенно снизить потребляемую мощность, но при этом ограничен частотный диапазон и нельзя XTAL2 использовать для внешней синхронизации.

При использовании резонаторов максимальная частота равна 8 МГц, если CKOPT – незапрограммирован, и 16 МГц, если CKOPT- запрограммирован. C1 и C2 должны быть всегда равны независимо от использования кварцевого или керамического резонатора. Оптимальное значение емкостей конденсаторов зависит от используемого кварцевого или керамического резонатора, от значения паразитной емкости и от окружающего уровня электромагнитного шума. Рекомендации по выбору номиналов конденсаторов приведены в таблице 8. Для керамических резонаторов необходимо использовать конденсаторы с номиналом, рекомендуемым производителем.

Рисунок 11. Соединения Кварцевого генератора



Генератор может работать в трех различных режимах, каждый оптимизированный для определенного диапазона частоты. Рабочий режим выбирается предохранителями CKSEL3.. 1 как показано в **Таблице 4.**

CKOPT	CKSEL3..1	Frequency Range(MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
1	101(1)	0.4 - 0.9	–
1	110	0.9 - 3.0	12 - 22
1	111	3.0 - 8.0	12 - 22
0	101, 110, 111	1.0 ≤	12 - 22

(1)Данная опция должна задаваться только при использовании керамического резонатора, а не кварцевого.

Конфигурационные биты CKSEL0 совместно с битами SUT1..0 выбирают время старта в соответствии с таблицей 5.

Таблица 5 – Временная задержка при запуске для различных настроек кварцевого генератора

CKSEL0	SUT1..0	Длительность задержки при выходе из режима выключения и экономичного режима	Дополнительная задержка после сброса (VCC= 5.0В)	Рекомендуемая область применения
0	00	258 CK(1)	4.1 мс	Керамический резонатор, быстро нарастающее питание
0	01	258 CK(1)	65 мс	Керамический резонатор, медленно нарастающее питание
0	10	1K CK(2)	–	Керамический резонатор, детектор питания (BOD) включен
0	11	1K CK(2)	4.1 мс	Керамический резонатор, быстро нарастающее питание
1	00	1K CK(2)	65 мс	Керамический резонатор, медленно нарастающее питание
1	01	16K CK	–	Кварцевый генератор, детектор питания (BOD) включен
1	10	16K CK	4.1 мс	Кварцевый резонатор, быстро нарастающее питание
1	11	16K CK	65 мс	Кварцевый резонатор, медленно нарастающее питание

Прим.:
Данные опции допускается использовать, только если микроконтроллер работает не на частоте близкой к максимальной рабочей, а также, если стабильность частоты при старте не важна для данного приложения. Данные опции не приемлемы при использовании кварцевых резонаторов.
Данные опции реализованы для использования керамических резонаторов и гарантируют стабильность частоты после запуска. При данных установках допускается использовать кварцевый резонатор, но при условии, что рабочая частота микроконтроллера меньше максимальной, а также, если стабильность частоты во время запуска не важна для данного приложения.

Низкочастотный кварцевый генератор

Для использования часового кварцевого резонатора 32.768 кГц в качестве источника синхронизации необходимо выбрать низкочастотный кварцевый генератор путем установки конфигурационных бит CKSEL равными “1001”. Подключение кварцевого резонатора показано на рисунке 11. Путем программирования конфигурационного бита пользователь может разрешить подключение встроенных конденсаторов к выводам XTAL1 и XTAL2, тем самым исключая необходимость применения внешних конденсаторов. Внутренние конденсаторы имеют номинал 36 пФ. После выбора данного генератора, длительности задержек при старте определяются конфигурационными битами SUT как показано в таблице 6.

Таблица 6 – Длительности задержек при старте для низкочастотного кварцевого резонатора

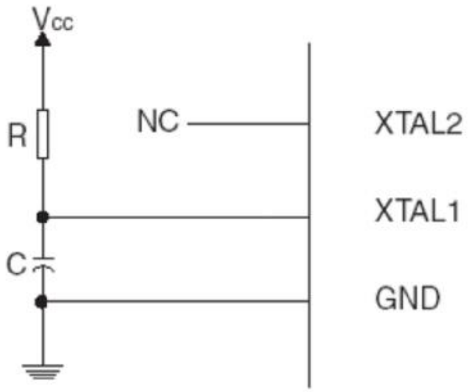
SUT1..0	Длительность задержки при выходе из режима выключения и экономичного режима	Дополнительная задержка после сброса (VCC= 5.0В)	Рекомендуемая область применения
00	1K CK(1)	4.1 мс	Быстро нарастающее питание или включен детектор питания BOD
01	1K CK(1)	65 мс	Медленно нарастающее питание
10	32K CK	65 мс	Стабильная частота при старте
11	Зарезервировано		

1. Данные опции необходимо использовать, если стабильность частоты при старте не важна для приложения.

Внешний RC-генератор

Для приложений некритичных к стабильности временных характеристик в качестве источника синхронизации может использоваться внешняя RC-цепь, подключение которой показано на рисунке 12. Тактовая частота грубо определяется выражением f = 1/(3RC). Номинал конденсатора C должен быть не менее 22 пФ. Путем программирования конфигурационного бита CKOPT пользователь может разрешить подключение внутреннего конденсатора 36 пФ между XTAL1 и GND, тем самым исключая необходимость применения внешнего конденсатора.

Рисунок 12. Внешняя Конфигурация RC



Генератор может работать в четырех различных режимах, каждый из которых ориентирован на специфический частотный диапазон. Рабочий режим выбирается конфигурационными битами CKSEL3..0 (см. табл. 7).

Таблица 7 – Рабочие режимы внешнего RC-генератора

CKSEL3..0	Частотный диапазон, МГц
0101	- 0.9
0110	0.9 - 3.0
0111	3.0 - 8.0
1000	8.0 - 12.0

После разрешения работы данного генератора длительность задержки при старте определяется установками конфигурационных бит (см. табл. 8).

Таблица 8 – Длительность задержек при старте после выбора внешнего RC-генератора

SUT1..0	Длительность задержки при выходе из режима выключения и экономичного режима	Дополнительная задержка после сброса (VCC= 5.0В)	Рекомендуемая область применения
00	18 CK(1)	-	Включен детектор питания BOD
01	18 CK	4.1 мс	Быстро нарастающее питание
10	18 CK	65 мс	Медленно нарастающее питание
11	6 CK (1)	4.1 мс	Быстро нарастающее питание или включенный детектор питания BOD

1. Данная опция не должна использоваться на тактовых частотах близких к максимальной.

Встроенный калиброванный RC-генератор

Встроенный калиброванный RC-генератор формирует фиксированные тактовые частоты 1.0, 2.0, 4.0 или 8.0 МГц. Данные значения частот являются номинальными и определены для напряжения питания 5В при 25?С. Одна из этих частот может быть выбрана в качестве тактовой, если запрограммировать конфигурационные биты CKSEL в соответствии с таблицей 9. После выбора микроконтроллер будет работать без внешних компонентов.

Конфигурационный бит CKOPT должен быть всегда незапрограммированным, если используется внутренний RC-генератор. В процессе сброса калибровочный байт аппаратно записывается регистр OSCCAL, тем самым автоматически выполняя калибровку RC-генератора. При питании 5В, температуре 25?С и выбранной частоте генератора 1.0 МГц данный метод калибровки обеспечивает погрешность генерации частоты не хуже ± 3% от номинального значения. Использование методов калибровки во время работы микроконтроллера позволяет достичь точности ± 1% при любой заданной температуре и напряжении VCC (см. рекомендации по применению www.atmel.com/avr). При использовании данного генератора в качестве тактового генератор сторожевого таймера также останется использоваться для тактирования сторожевого таймера и для задания длительности задержки при сбросе. Более подробная информация о предварительно запрограммированном калибровочном значении приведена в разделе “Калибровочный байт”.

Таблица 9 – Режимы встроенного калиброванного RC-генератора

CKSEL3..0	Номинальная частота, МГц
0001(1)	1.0
0010	2.0
0011	4.0
0100	8.0

Прим.: 1. Микроконтроллер поставляется с данной установкой.

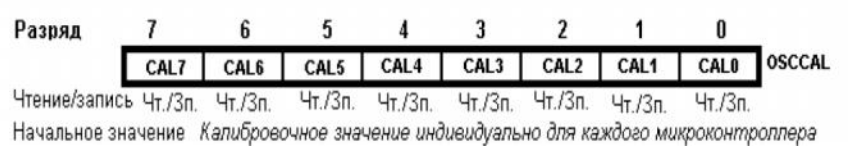
После выбора данного генератора длительность задержки при запуске микроконтроллера определяется установками конфигурационных бит SUT (см. табл. 10). Выводы XTAL1 и XTAL2 должны быть оставлены неподключенными (NC).

Таблица 10 – Длительности задержек при запуске с различными настройками встроенного калиброванного RC-генератора

SUT1..0	Длительность задержки при выходе из режима выключения и экономичного режима	Дополнительная задержка после сброса (VCC= 5.0В)	Рекомендуемые условия для применения
00	6 CK	-	Включен детектор питания BOD
01	6 CK	4.1 мс	Быстро нарастающее питание
10(1)	6 CK	65 мс	Медленно нарастающее питание
11	Зарезервировано		

Прим.: 1. Микроконтроллер поставляется с данной установкой.

Регистр калибровки генератора – OSCCAL



Разряды 7..0 – CAL7..0: Калибровочное значение для генератора
Запись значения калибровочного байта в данный регистр приведет к подстройке генератора на номинальную частоту. В процессе сброса калибровочное значение для частоты 1МГц (расположен в старшем байте строки сигнатуры) автоматически записывается в регистр OSCCAL. Если встроенный RC-генератор используется на других частотах, то калибровочный байт необходимо записывать программно. Для этого необходимо с помощью программатора считать значение калибровочного байта, затем сохранить его значение во флэш-память или EEPROM. После этого, калибровочное значение может быть считано программно, а затем записано в регистр OSCCAL. Если в регистр OSCCAL записать ноль, то выбирается минимальная частота. Запись ненулевого значения приводит к повышению частоты генератора. Запись $FF – к выбору максимальной частоты. Калиброванный генератор используется для синхронизации доступа к EEPROM и флэш- памяти. Во время выполнения записи в EEPROM или во флэш-память не следует выполнять калибровку на частоту выше на 10% от номинальной. В противном случае, запись в EEPROM или во флэш-память может быть некорректной. Обратите внимание, что генератор откалиброван отдельно на частоты 1.0, 2.0, 4.0 или 8.0 МГц. Результат подстройки при записи различных значений приведен в таблице 11.

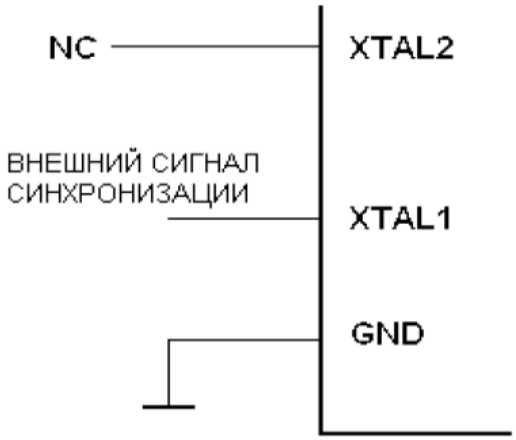
Таблица 11 – Диапазон частот встроенного RC-генератора

Значение OSCCAL	Минимальная частота в процентах от номинальной, %	Максимальная частота в процентах от номинальной, %
$00	50	100
$7F	75	150
$FF	100	200

Внешняя синхронизация

Если необходимо тактировать микроконтроллер от внешнего источника, то его необходимо подключить к выводу XTAL1 (см. рисунок 21). В этом случае внешняя синхронизация должна быть разрешена записью в конфигурационные биты CKSEL значения “0000”. Если запрограммировать конфигурационный бит CKOPT, то между XTAL1 и GND будет подключен внутренний конденсатор номиналом 36 пФ.

Рисунок 13 – Схема подключения внешнего источника синхронизации



После выбора данного источника синхронизации длительность задержки при запуске определяется конфигурационными битами SUT как показано в таблице 12.

Таблица 12 – Длительность задержки при запуске при выборе внешней синхронизации

SUT1..0	Длительность задержки при выходе из режима выключения и экономичного режима	Дополнительная задержка после сброса (VCC= 5.0В)	Рекомендуемые условия для применения
00	6 CK	-	Включен детектор питания BOD
01	6 CK	4.1 мс	Быстро нарастающее питание
10(1)	6 CK	65 мс	Медленно нарастающее питание
11	Зарезервировано		

После подключения внешнего тактового источника необходимо избегать внезапных изменений его частоты для гарантирования стабильности работы микроконтроллера. Если на следующем такте частота изменится более чем на 2% по сравнению с предыдущим, то поведение микроконтроллера может стать непредсказуемым. Данный механизм реализован для гарантирования нахождения микроконтроллера в состоянии сброса в процессе таких изменений тактовой частоты.

Генератор таймер-счетчика

Выводы генератора таймера-счетчика TOSC1 и TOSC2 предназначены для непосредственного подключения кварцевого резонатора. В этом случае не требуются внешние конденсаторы. Генератор оптимизирован для совместной работы с часовым кварцевым резонатором 32.768 кГц. Подключение внешнего тактового источника к выводу TOSC1 не рекомендуется.

Управление энергопотреблением и режимы сна

Использование режимов сна позволяет отключать неиспользуемые модули микроконтроллера,тем самым уменьшая потребляемую мощность. Микроконтроллер поддерживает несколько режимов сна, позволяющих программисту оптимизировать энергопотребление под требования приложения.
Для перевода микроконтроллера в один из пяти режимов сна необходимо предварительно установить бит SE в регистре MCUCR, а затем выполнить инструкцию SLEEP. Биты SM2, SM1 и SM0 регистра MCUCR задают в какой именно режим будет переведен микроконтроллер (холостой ход "Idle", уменьшение шумов АЦП "ADC Noise Reduction", выключение "Power-down", экономичный "Power-save", дежурный "Standby") после выполнения команды SLEEP (см. табл. 13). Выход из режима сна происходит при возникновении разрешенного прерывания. В этом случае, помимо времени старта микроконтроллер приостанавливается на 4 машинных цикла, выполняет процедуру обработки прерывания и продолжает выполнять команды следующие за SLEEP. Содержимое файла регистров и статического ОЗУ остается неизменным после выхода из режима сна. Если во время действия режима сна возникает условие сброса, то микроконтроллер пробуждается и исполняет код программы по вектору сброса.
Отметьте, что Расширенный Дежурный режим, существующий во многих других MCU AVR, был удален в ATmega8, поскольку TOSC и вводы XTAL совместно используют те же самые физические контакты.

На рисунке 10 представлены различные системы синхронизации микроконтроллера ATmega8 и их распределение. Данный рисунок может быть полезным при выборе соответствующего режима сна.

Регистр управления микроконтроллером – MCUCR

Регистр управления микроконтроллером содержит биты управления энергопотреблением.

Bit	7	6	5	4	3	2	1	0	
	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Разряд 5 – SE: Разрешение перевода в режим сна
В бит SE должна быть записана лог. 1, когда необходимо микроконтроллер перевести в режим сна командой SLEEP. Во избежание незапланированного программистом перевода микроконтроллера в режим сна рекомендуется устанавливать этот бит непосредственно перед выполнением инструкции SLEEP и сбрасывать сразу после пробуждения.
Разряды 4..2 – SM2..0: Биты 2, 1 и 0 выбора режима сна
С помощью данных бит можно выбрать один из шести режимов сна в соответствии с таблицей 13.

Таблица 13 – Выбор режима сна

SM2	SM1	SM0	Наименование режима сна
0	0	0	Холостой ход
0	0	1	Уменьшение шумов АЦП
0	1	0	Выключение
0	1	1	Экономичный
1	0	0	Зарезервирован
1	0	1	Зарезервирован
1	1	0	Дежурный (1)

Прим. 1: Дежурный режим и расширенный дежурный режим доступны только при использовании внешних кварцевых или керамических резонаторов.

Режим холостого хода (Idle)

Если значение бит SM2..0 равно 000, то после выполнения инструкции SLEEP микроконтроллер переходит в режим холостого хода, в котором останавливается ЦПУ, но продолжают работу SPI, УСАПП, аналоговый компаратор, АЦП, двухпроводной интерфейс, таймеры-счетчики, сторожевой таймер и система прерываний. По сути, в данном режиме останавливается синхронизация ядра ЦПУ и флэш-памяти (clkCPU и clkFLASH), а остальная продолжает работу.

В режиме холостого хода допускается пробуждение от любого внешнего или внутреннего прерывания, например, при переполнении таймера или завершении передачи УСАППом. Если пробуждение по прерыванию аналогового компаратора не требуется, то аналоговый компаратор может быть отключен путем установки бита ACD в регистре управления и состояния аналогового компаратора ACSR. Это позволит уменьшить потребляемый ток в режиме холостого хода. Если разрешена работа АЦП, то преобразование автоматически запускается после перевода в данный режим.

Режим уменьшения шумов АЦП (ADC Noise Reduction)

Если значениям бит SM2..0 присвоить 001, то выполнение инструкции SLEEP приведет к переводу микроконтроллера в режим уменьшения шумов АЦП, в котором останавливается ЦПУ,но продолжают работу АЦП, внешние прерывания, наблюдение за адресом двухпроводной последовательного шины, таймер-счетчик 0 и сторожевой таймер (конечно, если были предварительно активизированы). Фактически в данном режиме прекращается синхронизация ввода-вывода (clkI/O), ядра ЦПУ (clkCPU) и флэш-памяти (clkFLASH), а остальная синхронизация продолжает работу.
В этом режиме создается более благоприятные условия для аналогово-цифрового преобразования с повышенной разрешающей способностью за счет снижения влияния шумов на результат измерения. Если разрешена работа АЦП, то преобразование автоматически запускается при переводе в данный режим. Выход из данного режима допускается не только при генерации запроса на прерывание по завершению преобразования АЦП, но и при внешнем сбросе, сбросе по сторожевому таймеру, сбросе при недопустимом снижении питания,прерывании при обнаружении установленного адреса на двухпроводной последовательной шине,прерывании по таймеру-счетчику 0, прерывании по готовности SPM/EEPROM, прерывании по внешнему уровню на выводах INT0 или INT1.

Режим выключения (Power-down)

Если SM2..0 = 010, то выполнение команды SLEEP означает перевод микроконтроллера в режим выключения. В данном режиме прекращает работу внешний генератор, но в действии остаются внешние прерывания, наблюдение за адресом на двухпроводной последовательной шине и сторожевой таймер (при условии, что они активизированы). Выход из данного режима возможен только по внешнему сбросу, сбросу сторожевым таймером, сбросу супервизором питания, прерывании по обнаружении установленного адреса на двухпроводной последовательной шине, прерывании по внешнему уровню на выводах INT0:1 . В данном режиме фактически отключена генерация всех тактовых частот,поэтому дальнейшая работа модулей продолжается только в асинхронном режиме.
Обратите внимание, что если для выхода из прерывания используется прерывание по установке заданного уровня на внешнем входе, то выход из режима сна возможен только если этот уровень присутствует в течение определенного времени (см. также “Внешние прерывания”).
Выход из режима выключения сопровождается задержкой с момента выполнения условия прерывания до эффективного пробуждения. Данная задержка позволяет перезапустить синхронизацию и дождаться стабильности ее работы. Период пробуждения определяется теми же самыми Предохранителями CKSEL, которые определяют период Тайм-аута Сброса, как описано в “Источниках Часов” на странице 26.

Экономичный режим (Power-save)

Если установить значения бит SM2..0 равным 011, то действие команды SLEEP приведет к переводу микроконтроллера в экономичный режим. Данный режим идентичен режиму выключения за некоторыми исключениями: Если таймер-счетчик 0 тактируется асинхронно, т.е. установлен бит AS0 в регистре ASSR, то таймер-счетчик 0 в режиме сна продолжит работу. Выход из режима сна возможен как по переполнению таймера, так и при выполнении условия сравнения, если соответствующее прерывание для таймера-счетчика разрешено в регистре TIMSK, а также установлен бит общего разрешения прерываний в регистре SREG.
Если для асинхронного таймера НЕ включено асинхронное тактирование, то рекомендуется использовать режим выключения вместо экономичного, т.к.

содержимое регистров асинхронного таймера должно рассматриваться как неопределенное после выхода из экономичного режима, в котором значение AS0 было равно 0.
В данном режиме сна останавливаются все тактовые источники за исключением асинхронных (clkASY), работающих только совместно с асинхронными модулями, в т.ч. таймер-счетчик 0 с разрешенной опцией асинхронного тактирования.

Дежурный режим (Standby)

После установки значения SM2..0 = 110 и выбора опции тактирования от внешнего кварцевого или керамического резонатора выполнение инструкции SLEEP приводит к переходу микроконтроллера в дежурный режим. Данный режим идентичен режиму выключению за исключением того, что генератор продолжает свою работу. Из дежурного режима микроконтроллер выходит за 6 машинных циклов.

Таблица 14 – Активные тактируемые модули и источники пробуждения в различных режимах сна

Наименование режима сна	Тактируемые модули микроконтроллера					Активные генераторы		Источник пробуждения					
	clkCPU (ЦПУ)	clkFLASH (флэш -память)	clkIO (ввод-вывод)	clkADC (АЦП)	clkASY (Асин. модули.)	Основной тактовый	Генератор таймера	INT1:0	Набл. адреса TWI	Таймер 0	Готовность SPM/ EEPROM	АЦП	Др. ввод-вывод
Холостой ход			x	x	x	x	x(2)	x	x	x	x	x	x
Уменьшение шумов АЦП				x	x	x	x(2)	x(3)	x	x	x	x	
Выключение								x(3)	x				
Экономичный					x(2)		x(2)	x(3)	x	x(2)			
Дежурный (1)						x		x(3)	x				

Прим.
1. В качестве внешнего тактового источника выбран кварцевый или керамический резонатор.
2. Если установлен бит AS0 в ASSR.
3. Только прерывание по уровню INT1 или INT0

Минимизация потребляемой мощности

В процессе оптимизации энергопотребления могут возникнуть некоторые проблемы. В общем случае необходимо по возможности максимально использовать режимы сна, а собственно режим сна необходимо выбрать исходя из поддержки только необходимых функций. Все неиспользуемые функции должны быть отключены. В частности, для следующих модулей в целях достижения наименьше возможного энергопотребления должны быть учтены некоторые рекомендации

Аналогово-цифровой преобразователь

Если АЦП был активизирован, то он останется активным и во всех режимах сна. Для снижения мощности рекомендуется отключать АЦП перед переводом в режим сна. Если АЦП был отключен, а затем снова включен, то следующее преобразование будет расширенным (см. “Аналогово-цифровой преобразователь").

Аналоговый компаратор

Перед входом в режим холостого хода аналоговый компаратор необходимо выключить, если он не используется. Перед входом в режим уменьшения шумов АЦП аналоговый компаратор должен быть отключен. При входе в другие режимы сна аналоговый компаратор отключается автоматически. Однако, если к неинвертирующему входу аналогового компаратора выбрано подключение встроенного источника опорного напряжения, то перед входом в любой режим сна аналоговый компаратор необходимо отключать. В противном случае встроенный источник опорного напряжения останется включенным независимо от режима сна (см. также “Аналоговый компаратор”).

Супервизор питания

Если нет необходимости использовать супервизор питания, то данный модуль может быть выключен. Если супервизор питания активизирован конфигурационным битом BODEN, то он также останется активным и во всех режимах сна и, следовательно, будет постоянно потреблять мощность. При организации режимов глубокого сна отключение данного модуля позволит существенно уменьшит потребляемый ток (см. также "Супервизор питания").

Встроенный источник опорного напряжения

Работа встроенного источника опорного напряжения разрешается, если необходимо использовать супервизор питания, аналоговый компаратор или АЦП. Если данные модули будут отключены, то встроенный ИОН также будет отключен и не будет потреблять мощность. При возобновлении его работы программист должен учесть задержку на установление выходного напряжения ИОН перед его использованием. Если ИОН остается включенным в режиме сна, то его можно использовать сразу после пробуждения (см. также “Встроенный источник опорного напряжения ” для уточнения времени его запуска).

Сторожевой таймер

Если нет необходимости в использовании сторожевого таймера, то данный модуль должен быть отключен. Если разрешить работу сторожевого таймера, то он останется активным во всех режимах сна и, следовательно, будет потреблять мощность. Если требуются режимы глубокого сна, то данная опция позволит существенно снизить общий ток (см. также “Сторожевой таймер”).

Линии портов ввода-вывода

Перед переводом в режим сна все линии портов ввода-вывода должны быть настроены с учетом потребления минимальной мощности. Основное внимание следует уделить отсутствию резистивных нагрузок на выводах. В режимах сна, где отключена синхронизация ввода-вывода (clkI/O) и АЦП (clkADC), входные буферы микроконтроллера отключены. Этим гарантируется отсутствие энергопотребления неиспользуемой в режиме сна входной логикой. В некоторых случаях входная логика необходима для определения условия пробуждения и в этом случае должна быть активной (см. также “Разрешение цифрового ввода и режимы сна ”). Если работа входного буфера разрешена, а входной сигнал оказался отключенным или имеет уровень близкий к VCC/2, то этот входной буфер будет потреблять повышенную мощность.

Сброс микроконтроллера

В процессе сброса во все регистры ввода-вывода записываются их начальные значения и выполнение программы начинается с вектора сброса. По вектору сброса должна хранится инструкция абсолютного перехода JMP на метку процедуры обработки сброса. Если в программе не используются источники прерывания, то векторы прерываний не используются, а зарезервированные под них ячейки памяти могут использоваться для равномерного расположения кода программы. Имеется также случай, когда вектор сброса расположен в секции прикладной программы, а векторы прерываний находятся в загрузочном секторе или наоборот. На рисунке 14 показана схема организации логики сброса. В таблице 15 приведены электрические характеристики схемы сброса.

После прекращения действия всех источников сброса вступает в силу счетчик задержки, продлевающий внутренний сброс. Данная последовательность требуется для гарантирования запуска микроконтроллера при достижении напряжением питания стабильного уровня. Длительность задержки при старте определяется конфигурационными битами CKSEL. Различные варианты установок длительностей задержек представлены в разделе “Источники синхронизации”стр.26.

Источники сброса

ATmega8 имеет четыре источника сброса:

- Сброс при подаче питания. Микроконтроллер переходит в состояние сброса, если напряжение питания ниже порога сброса при подаче питания (VPOT).
- Внешний сброс. Микроконтроллер переходит в состояние сброса, если на вывод RESET подать низкий логический уровень на время дольше, чем минимальная длительность импульса сброса.
- Сброс по сторожевому таймеру. Если разрешена работа сторожевого таймера и стек период его срабатывания, то микроконтроллер сбрасывается.
- Сброс при снижении питания. Микроконтроллер сбрасывается, если напряжение питания VCC становится ниже порогового значения (VBOT) и разрешена работа схемы контроля питания BOD.

Рисунок 14. Логика сброса

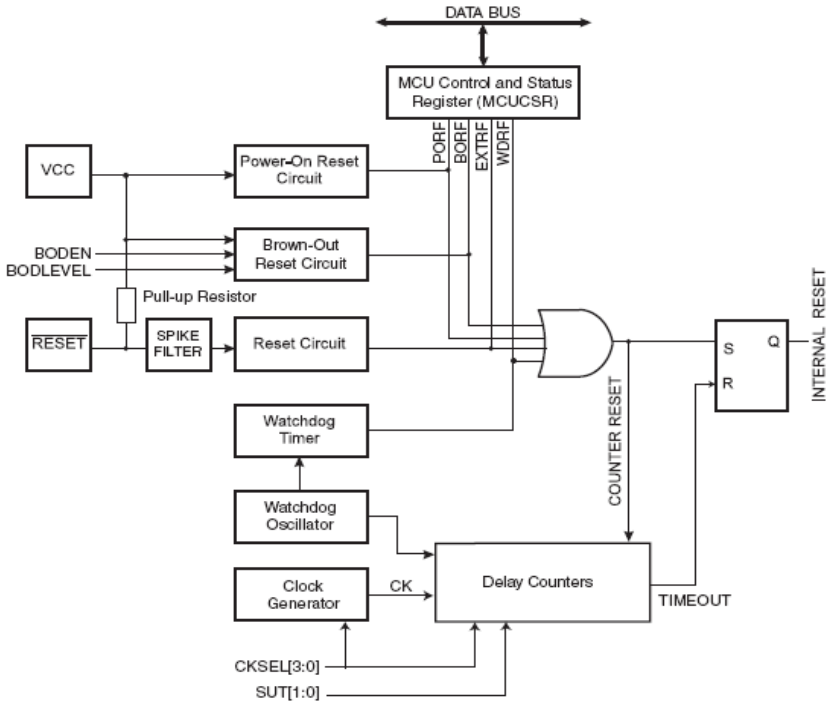


Таблица 15. Характеристики сброса

Обозн.	Параметр	Условие	Мин.	Тип.	Макс.	Единица измерения
VPOT	Пороговое напряжение сброса при повышении питания			1.4	2.3	**В**
	Пороговое напряжение сброса при снижении питания (1)			1.3	2.3	**В**
VRST	Пороговый уровень сброса на выводе RESET		0.2 Vcc		0.9	**Vcc**
tRST	Минимальная длительность импульса сброса на выводе RESET				1.5	мкс
VBOT	Порог напряжения сброса схемы контроля питания BOD(2)	BODLEVEL = 1	2.4	2.6	2.9	**В**
		BODLEVEL = 0	3.7	4.0	4.5	
tBOD	Минимальная длительность снижения напряжения для срабатывания схемы контроля питания	BODLEVEL = 1		2		мкс
		BODLEVEL = 0		2		мкс
VHYST	Ширина петли гистерезиса схемы контроля питания			130		мВ

Примечания:
1. Сброс при подаче питания не будет работать, если напряжение питания ниже VPOT.
2. У некоторых микроконтроллеров VBOT может быть ниже минимального рабочего напряжения. Данные микроконтроллеры на стадии производства испытываются при VCC = VBOT. Этим гарантируется то, что сброс микроконтроллера будет выполнен раньше, чем произойдет снижение питания микроконтроллера на недопустимый для его корректной работы уровень. Испытание ATmega8L выполнено при BODLEVEL=1, а ATmega8 при BODLEVEL=0. Установка BODLEVEL=1 не применима для ATmega8.

Сброс при подаче питания

Импульс сброса при подаче питания (POR) генерируется встроенной схемой. Пороговый уровень сброса приведен в таблице 15. POR инициируется всякий раз, когда VCC ниже порогового уровня. Схема POR может использоваться для запуска микроконтроллера, а также для выявления нарушения режима питания.

схема сброса по включению питания (POR), гарантирует, что устройство сбрасывается от включения питания. Достижение порогового напряжения сброса включения питания вызывает счетчик задержки, который определяет, сколько времени устройство сохраняется в СБРОСЕ после повышения VCC. Сигнал СБРОСА активируется снова, без любой задержки, когда VCC уменьшается ниже уровня обнаружения.

Рисунок 15. Запуск микроконтроллера (RESET соединен с VCC).

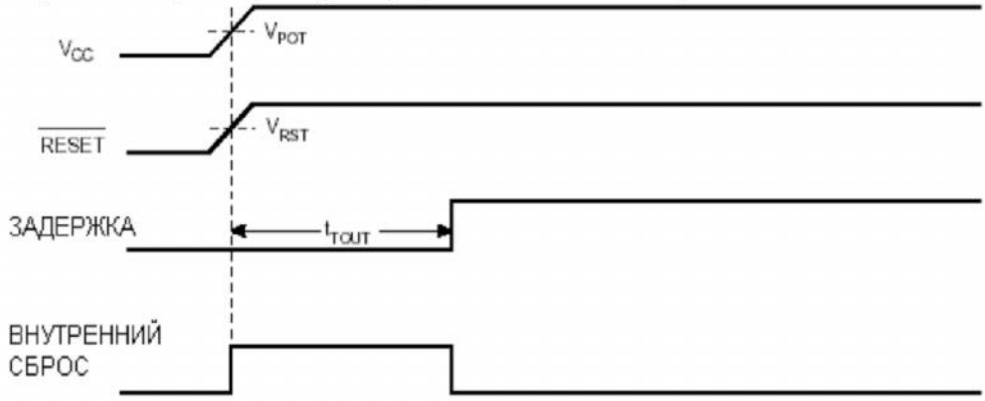
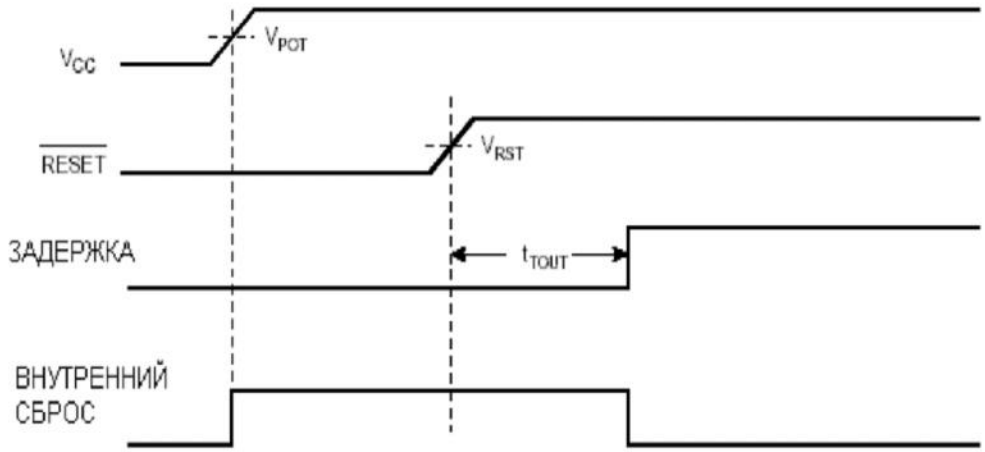


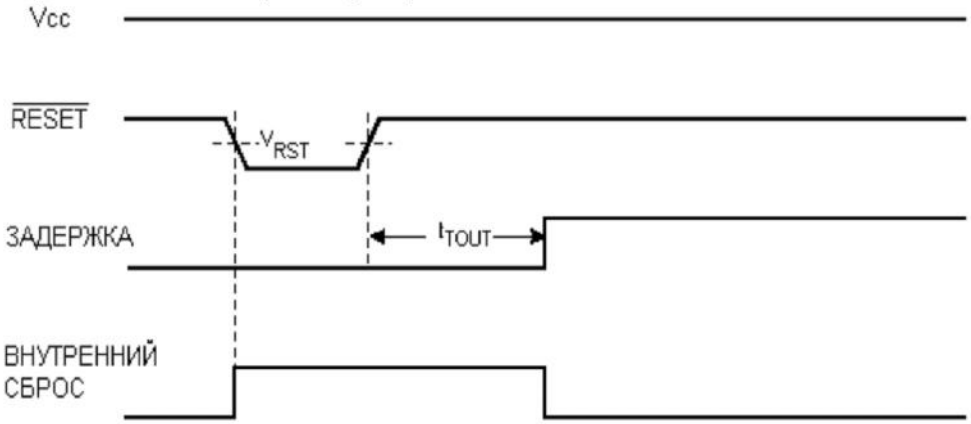
Рисунок 16. Запуск микроконтроллера (RESET управляется внешне).



Внешний сброс

Внешний сброс генерируется, если на вход RESET подать низкий уровень. Если подать импульс сброса длительностью более tRST (см. табл. 15), то будет генерирован сброс, даже если синхронизация не запущена. При подаче импульса сброса длительностью менее tRST сброс не гарантируется. Если сигнал на выводе RESET достигает порогового напряжения сброса VRST на его положительном фронте, то запускается счетчик задержки и микроконтроллер начнет работу только по истечении периода tTOUT.

Рисунок 17. Внешний Сброс во время работы



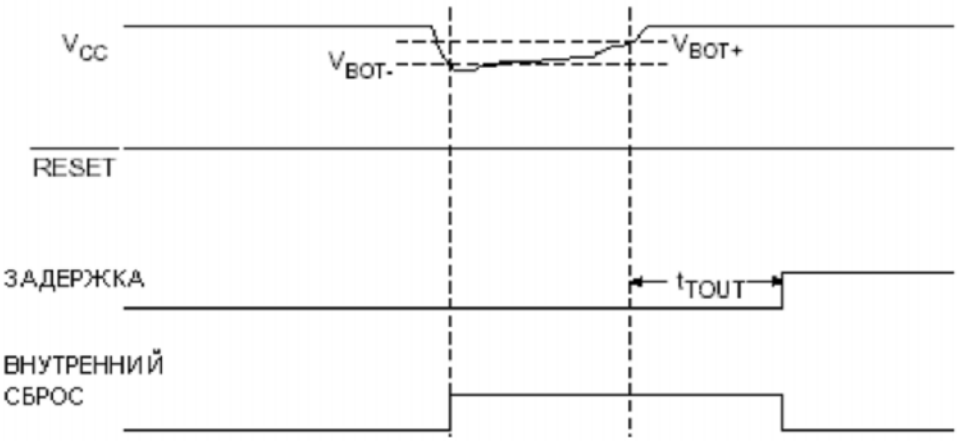
Контроль напряжения питания

ATmega8 содержит встроенную схему контроля питания (BOD), которая выполняет сравнение уровня VCC с фиксированным пороговым значением. Порог срабатывания схемы BOD может выбираться с помощью конфигурационного бита BODLEVEL. Порог равен 2.7В, когда BODLEVEL незапрограммирован, или 4.0В, когда BODLEVEL запрограммирован. Для исключения автоколебательного режима схема BOD характеризуется гистерезисом. С учетом гистерезиса результирующие пороги срабатывания следующие: VBOT+ = VBOT + VHYST/2 и VBOT- = VBOT - VHYST/2.

Схема BOD может быть включена или отключена с помощью конфигурационного бита BODEN. Если разрешена работа BOD (BODEN запрограммирован) и уровень VCC снизился ниже порога срабатывания (VBOT- на рисунке 18), то схема BOD переводит микроконтроллер в состояние сброса. Когда VCC достигает значения выше порога срабатывания (VBOT+ на рисунке 18), то запускается счетчик задержки и микроконтроллер начнет работу по истечении времени tTOUT.

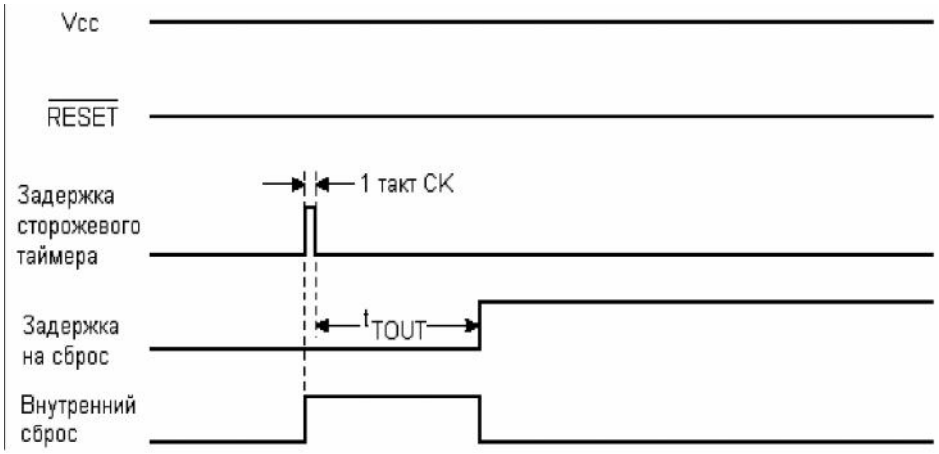
Схема BOD реагирует на снижение VCC, если напряжение остается меньшим порога срабатывания дольше чем период времени tBOD (см. табл. 15).

Рисунок 18. Сброс микроконтроллера схемой контроля питания



По истечении периода переполнения сторожевого таймера генерируется короткий импульс сброса длительностью равной одному периоду системной синхронизации (1 CK). Падающим фронтом этого импульса запускается счетчик задержки tTOUT.

Рисунок 19. Сброс сторожевым таймером



Регистр управления и статуса микроконтроллера – MCUCSR

В регистре управления и статуса микроконтроллера хранится информация об источнике, который вызвал сброс микроконтроллера.

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

Б Бит 7.. 4– Res: Зарезервированные биты.
Эти биты - зарезервированные биты в ATmega8 и всегда читают как нуль.

3 – WDRF: Флаг индикации сброса сторожевым таймером
Данный бит устанавливается после сброса сторожевым таймером. Данный бит сбрасывается при подаче питания или путем записи лог. 0 в данный флаг.

– BORF: Флаг Сброса снижения напряжения
Данный флаг принимает единичное состояние, если схема контроля напряжения питания перевела микроконтроллер в состояние сброса. Данный бит сбрасывается при подаче питания и путем записи лог. 0 в данный флаг.

– EXTRF: Флаг Внешнего Сброса
Данный бит устанавливается при возникновении внешнего сброса. Флаг сбрасывается при подаче питания или путем записи лог. 0 в данный флаг.

0 – PORF: Флаг Сброса включении питания
Данный флаг устанавливается, если сброс был инициирован подачей питания. Данный бит сбрасывается только путем записи лог. 0 в данный флаг.
Если флаги сброса необходимо использовать для определения причины сброса, то программист должен предусмотреть сброс значений флагов MCUCSR желательно сразу после опроса их значений. Если регистр очищается перед возникновением другого сброса, то источник данного сброса может быть найден среди флагов сброса.

Встроенный источник опорного напряжения

ATmega8 содержит встроенный источник опорного напряжения (ИОН). ИОН используется схемой контроля питания и может быть подключен ко входу аналогового компаратора или к АЦП. Опорное напряжение АЦП 2.56В генерируется встроенным ИОН.

Сигналы разрешения и длительность запуска ИОН

ИОН характеризуется задержкой при включении, которая может оказать негативное влияние, если не будет учтена. Длительность задержки приведена в таблице 20. Для оптимизации энергопотребления ИОН не всегда находится во включенном состоянии. ИОН остается во включенном состоянии в следующих случаях:
1. Когда разрешена работа схемы контроля питания BOD (запрограммирован конфигурационный бит BODEN).
2. Если ИОН подключен ко входу аналогового компаратора (установлен бит ACBG в ACSR).
3. Если разрешена работа АЦП.

Следовательно, когда работа BOD запрещена и установлен бит ACBG или разрешена работа АЦП, программист должен предусмотреть задержку на время запуска ИОН перед использованием выходных данных аналогового компаратора или АЦП. В целях снижения потребляемой мощности в режиме выключения (Power-down) программист должен избежать приведенных выше трех условий для гарантирования, что ИОН будет отключен после перевода микроконтроллера в режим выключения.

Таблица 16. Характеристики встроенного ИОН

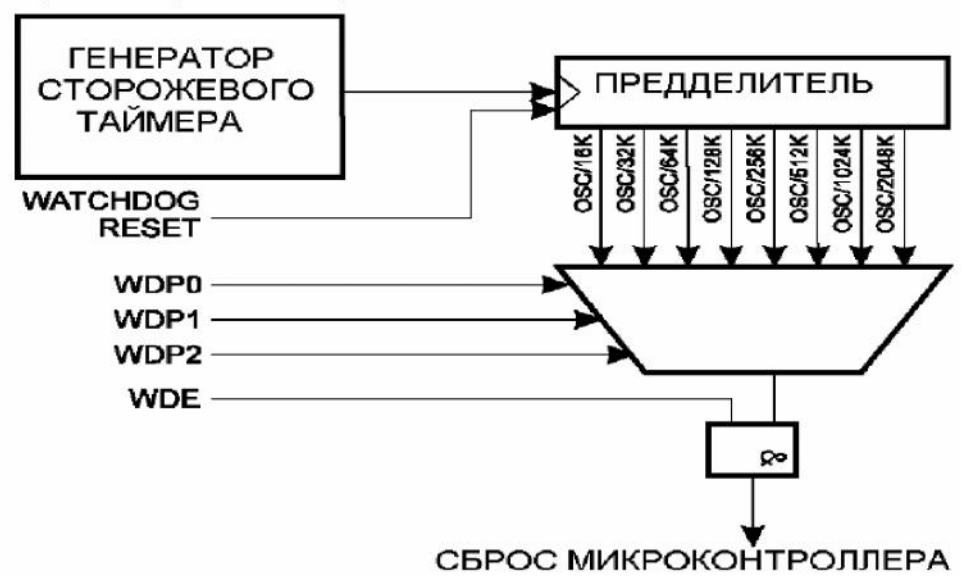
Обозн.	Параметр	Мин.	Тип.	Макс.	Ед.изм.
VBG	Напряжение ИОН	1.15	1.23	1.40	В
tBG	Длительность запуска ИОН		40	70	мкс
IBG	Потребляемый ток ИОНом		10		мкА

Сторожевой таймер

Сторожевой таймер тактируется от отдельного встроенного генератора частотой 1 МГц. Данное значение типично для напряжения питания VCC = 5В. См. данные характеристики для типичных значений на других уровнях VCC.
Период переполнения сторожевого таймера можно задавать путем управления предделителем (см. табл. 17). Инструкция WDR выполняет сброс сторожевого таймера. Сторожевой таймер также сбрасывается при выключении или во время сброса микроконтроллера. Период переполнения определяют восемь различных коэффициентов деления предделителя. Если инструкция сброса сторожевого таймера WDR не выполняется в течение времени равного периоду переполнения сторожевого таймера, то ATmega8 сбрасывается и начинает выполнение программы по вектору сброса.

Чтобы предотвратить неумышленное отключение Сторожевого таймера, специальная последовательность должна сопровождаться, когда Сторожевой таймер отключается. См. описание для Сторожевого Таймера Регистра команд для деталей.

Рисунок 20. Сторожевой Таймер



Регистр управления сторожевого таймера – WDTCR

Разряд	7	6	5	4	3	2	1	0	
	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0	WDTCR
Чтение/Запись	Чт.	Чт.	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Нач. значение	0	0	0	0	0	0	0	0	

– Res: Зарезервированные Биты
Данные разряды являются зарезервированными и всегда считываются как 0.

– WDCE: Разрешение изменения сторожевого таймера
Данный бит необходимо установить непосредственно перед записью лог. 0 в бит WDE. В противном случае запретить работу сторожевого таймере невозможно. После записи в данный бит лог. 1 он автоматически аппаратно сбросится по истечении четырех тактов синхронизации микроконтроллера. На уровнях безопасности 1 и 2 данный бит необходимо устанавливать также перед изменением настроек предделителя. См. “Временные последовательности изменения конфигурации сторожевого таймера”стр.45.

– WDE: Разрешение сторожевого таймера

Работа сторожевого таймера разрешается (запрещается) путем установки (сброса) бита WDE. Сбросить бит WDE возможно, только если предварительно установить бит WDCE. Для выключения сторожевого таймера необходимо выполнить следующую последовательность:

1. Записать лог. 1 в WDCE и WDE одной инструкцией. Лог. 1 должна быть записана в бит WDE, даже если до выполнения данной операции в нем уже была записана лог. 1.
2. В течение следующих четырех тактов записать лог. 0 в WDE, что приводит к отключению сторожевого таймера.

– WDP2, WDP1, WDP0: Выбор коэффициента деления

Биты WDP2, WDP1 и WDP0 задают коэффициент деления частоты генератора сторожевого таймера после разрешения работы последнего. Значения коэффициентов деления и соответствующих периодов переполнения приведены в табл. 17.

Таблица 17 – Настройка предделения генератора сторожевого таймера

WDP2	WDP1	WDP0	Количество тактов генератора сторожевого таймера	Типичное время переполнения при VCC = 3.0В	Типичное время переполнения при VCC = 5.0В
0	0	0	16K (16,384)	17.1 мс	16.3 мс
0	0	1	32K (32,768)	34.3 мс	32.5 мс
0	1	0	64K (65,536)	68.5 мс	65 мс
0	1	1	128K (131,072)	0.14 с	0.13 с
1	0	0	256K (262,144)	0.27 с	0.26 с
1	0	1	512K (524,288)	0.55 с	0.52 с
1	1	0	1,024K (1,048,576)	1.1 с	1.0 с
1	1	1	2,048K (2,097,152)	2.2 с	2.1 с

В следующих примерах приведены функции выключения сторожевого таймера на Ассемблере и Си. В примерах предполагается, что система прерываний настроена таким образом, чтобы во время выполнения функции не возникло прерывание (например, с помощью общего запрета прерываний инструкцией cli).

Временные последовательности изменения конфигурации сторожевого таймера

Последовательность изменения настройки сторожевого таймера плавно изменяется между уровнями безопасности. Ниже описаны процедуры изменения настроек для каждого из уровней.

Пример кода на Ассемблере

```
WDT_off:
; Запись лог. 1 в WDCE и WDE
in r16, WDTCR
ori r16, (1<<WDCE)|(1<<WDE)
out WDTCR, r16

; Выкл. сторожевого таймера
ldi r16, (0<<WDE)
out WDTCR, r16
ret
```

Пример кода на Си

```
void WDT_off(void)
{
/* reset WDT */
_WDR();

/* Запись лог. 1 в WDCE и WDE */
WDTCR |= (1<<WDCE) | (1<<WDE);

/* Выкл. сторожевого таймера */
WDTCR = 0x00;
 }
```

Уровень 1 безопасности (WDTON Незапрограммирован)

В данном режиме сторожевой таймер первоначально отключен. Его работа может быть разрешена путем записи лог. 1 в бит WDE без каких-либо ограничений. Временная последовательность должна быть соблюдена при изменении периода переполнения сторожевого таймера или выключении разрешенного сторожевого таймера. В данном случае должна быть выполнена следующая последовательность:

1. С помощью одной и той же инструкции записать лог. 1 в WDCE и WDE. Лог. 1 должна быть записана в WDE независимо от предыдущего значения бита WDE.
2. В течение следующих 4-х тактов одной инструкцией записать желаемое значение бит WDE и WDP, но со сброшенным значением бита WDCE.

Уровень 2 безопасности (WDTON Запрограммирован)

В данном режиме сторожевой таймер всегда включен а значение бита WDE всегда считывается как лог. 1. Временная последовательность должна быть соблюдена при изменении периода переполнения сторожевого таймера. При этом, должна быть выполнена следующая последовательность:

1. Записать лог. 1 в WDCE и WDE одной инструкцией. Не смотря на то, что WDE всегда установлен, для запуска временной последовательности в него все равно необходимо записывать лог. 1.
2. В течение следующих 4-х тактов записать желаемое значение WDP одной инструкцией при сброшенном значении бита WDCE. Записываемое значение в бит WDE не оказывает никакого влияния.

Прерывания

В данном разделе описывается специфика обработки прерываний, реализованная в ATmega8. Общее описание обработки прерываний приведено в разделе “Сброс и обработка прерываний”.стр.14

Векторы прерываний в ATmega8

Таблица 18. Векторы Сброса и Прерывания

№ вектора	Адрес памяти(2) Программ	Источник	Условие возникновения прерывания
1	0x000(1)	RESET	Внешний сброс, сброс при подаче питания, сброс при недопустимом снижении питания, сброс сторожевым таймером
2	0x001	INT0	Запрос на внешнее прерывание 0
3	0x002	INT1	Запрос на внешнее прерывание 1
4	0x003	TIMER2 COMP	Срабатывание компаратора таймера-счетчика 2
5	0x004	TIMER2 OVF	Переполнение таймера-счетчика 2
6	0x005	TIMER1 CAPT	Захват фронта таймером-счетчиком 1
7	0x006	TIMER1 COMPA	Срабатывание компаратора A таймера-счетчика 1
8	0x007	TIMER1 COMPB	Срабатывание компаратора B таймера-счетчика 1
9	0x008	TIMER1 OVF	Переполнение таймера-счетчика 1
10	0x009	TIMER0 OVF	Переполнение таймера-счетчика 0
11	0x00A	SPI, STC1	Завершение последовательной передачи интерфейсом SPI
12	0x00B	USART, RXC	Завершение приема УСАПП
13	0x00C	USART, UDRE	Регистр данных УСАПП свободен
14	0x00D	USART, TXC	Завершение передачи УСАПП
15	0x00E	ADC	Завершение преобразования АЦП
16	0x00F	EE_RDY	Готовность EEPROM
17	0x010	ANA_COMP	Аналоговый компаратор
18	0x011	TWI	Двухпроводный Последовательный интерфейс
19	0x012	SPM_RDY	Готовность записи Памяти программ

Прим.:

1. Если конфигурационный бит BOOTRST запрограммирован, то микроконтроллер выполняет переход на адрес сброса в загрузочном секторе, см. “ Самопрограммирование из сектора начальной загрузки с поддержкой чтения во время записи ”
2. Если установлен бит IVSEL в регистре MCUCR, то векторы прерываний перемещаются в начало загрузочного сектор флэш-памяти. В этом случае к адресу каждого вектора прерывания из таблицы прибавляется стартовый адрес загрузочного сектора флэш-памяти

В таблице19 показано расположение векторов сброса и прерываний в зависимости от различных установок BOOTRST и IVSEL. Если программа не использует прерывания, то она может быть размещена равномерно, используя ячейки с адресами векторов прерываний для хранения программного кода. Возможен также случай, когда вектор сброса располагается в секторе прикладной программы, а векторы прерываний – в загрузочном секторе или наоборот.

Таблица 19. Сброс и Размещение Векторов Прерывания

BOOTRST(1)	IVSEL	Адрес сброса	Начальный адрес векторов прерываний
1	0	0x000	0x000
1	1	0x000	Адрес сброса в загрузочном секторе +0x001
0	0	Адрес сброса в загрузочном секторе	0x001
0	1	Адрес сброса в загрузочном секторе	Адрес сброса в загрузочном секторе +0x001

Прим. : Адрес сброса загрузочного сектора показан в таблице 82. Для конфигурационного бита BOOTRST “1” означает незапрограммированное состояние, “0” - запрограммированное.

Ниже приведено большинство типичных и общих программных установок адресов сброса и векторов прерываний у ATmega8:

```
Адрес     Инструкция        Комментарий
$000 rjmp RESET ; Переход на обработку сброса
$001 rjmp EXT_INT0 ; Переход на обработку запроса IRQ0
$002 rjmp EXT_INT1 ; Переход на обработку запроса IRQ1
$003 rjmp TIM2_COMP ; Переход на обработку сравнения  Timer2
$004 rjmp TIM2_OVF ; Переход на обработку при переполнении Timer2
$005 rjmp TIM1_CAPT ; Переход на обработку при захвате фронта
Timer1
$006 rjmp TIM1_COMPA ; Переход на обработку при срабатывании
компаратора A Timer1
$007 rjmp TIM1_COMPB ; Переход на обработку при срабатывании
компаратора B Timer1
$008 rjmp TIM1_OVF ; Переход на обработку при переполнении Timer1
$009 rjmp TIM0_OVF ; Переход на обработку при переполнении Timer0
$00a rjmp SPI_STC ; Переход на обработку при завершении передачи
SPI
$00b rjmp USART_RXC ; Переход на обработку при завершении приема
УСАПП0
$00c rjmp USART_UDRE ; Переход на обработку при освобождении
регистра
$00d rjmp USART_TXC ; Переход на обработку при завершении передачи
$00e rjmp ADC ; Переход на обработку при завершении преобразования
 АЦП
$00f rjmp EE_RDY ; Переход на обработку при готовности EEPROM
$010 rjmp ANA_COMP ; Переход на обработку при срабатывании
аналогового компаратора
$011 rjmp TWSI ; Двухпроводный Последовательный интерфейс
$012 rjmp SPM_RDY ; Переход на обработку прерывания при Готовности
записи в память программ
$013 RESET: ldi r16,high(RAMEND); Начало основной программы
$014 out SPH,r16 ; Установка указателя стека в конце ОЗУ
$015 ldi r16,low(RAMEND)
$016 out SPL,r16
$017 sei ; Разрешение прерываний
$018 <instr> xxx
... ... ...
```

Если конфигурационный бит BOOTRST незапрограммирован, размер загрузочного сектора установлен 2 кбайт и бит IVSEL установлен в регистре GICR перед разрешением любого прерывания, то можно использовать следующий пример распределения программы по адресам векторов сброса и прерываний.

```
Адрес Инструкция Комментарий
$0000 RESET:ldi r16,high(RAMEND) ; Начало основной программы
$0001 out SPH,r16 ; Установка указателя стека в вершину ОЗУ
$0002 ldi r16,low(RAMEND)
$0003 out SPL,r16
$0004 sei ; Разрешение прерываний
$0005 <instr> xxx
;
.org $c01

$c01 jmp EXT_INT0 ; Переход на обработку прерывания IRQ0

$c02 jmp EXT_INT1 ; Переход на обработку прерывания IRQ1
... ... ... ;

$c12 jmp SPM_RDY ; Переход на обработку прерывания по готовности к
записи в память программ
```

Если конфигурационный бит BOOTRST запрограммирован и установлен размер загрузочного сектора 2 кбайт, то можно использовать следующий шаблон программы:

```
Адрес Инструкция Комментарий
.org $001
$001 rjmp EXT_INT0 ; Переход на обработку прерывания IRQ0
$002 rjmp EXT_INT1 ; ; Переход на обработку прерывания IRQ1
... ... ... ;
$012 rjmp SPM_RDY ; Переход на обработку прерывания по готовности к
записи в память программ
;
.org $c00
$c00 rjmp RESET ; обработка Reset
;
$c01 RESET:ldi r16,high(RAMEND); Начало основной программы
$c02 out SPH,r16 ; Установка Указателя  стека в вершину RAM

$c03 ldi r16,low(RAMEND)
$c04 out SPL,r16
$c05 sei ; Разрешение прерываний
$c06 <instr> xxx
```

Если конфигурационный бит BOOTRST запрограммирован, размер загрузочного сектора установлен 2 кбайт и бит IVSEL в регистре GICR установлен перед разрешение любого из прерываний, то распределение адресов в программе следующее:

```
Адрес Инструкция Комментарий
;
.org $c00
$c00 rjmp RESET ; Переход на обработку Reset
$c01 rjmp EXT_INT0 ; Переход на обработку IRQ0
$c02 rjmp EXT_INT1 ; Переход на обработку IRQ1
... ... ... ;
$c12 rjmp SPM_RDY ; Переход на обработку прерывания по готовности
записи в память программ
$c13 RESET: ldi r16,high(RAMEND); Начало основной программы
$c14 out SPH,r16 ; Установка Указателя  стека в вершину RAM
$c15 ldi r16,low(RAMEND)
$c16 out SPL,r16
$c17 sei ; Разрешение прерываний
$c18 <instr> xxx
```

Перемещение между секторами загрузочной и прикладной программы

Общий регистр управления прерываниями задает размещение таблицы векторов прерываний.

Общий Регистр Управления Прерыванием – GICR

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	–	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

1 – IVSEL: Выбор Вектора Прерывания

Если бит IVSEL сброшен (=0), то векторы прерываний размещаются в начале флэш-памяти. Если данный бит установлен (=1), то векторы прерываний перемещаются в начало загрузочного сектора флэш-памяти. Фактический адрес начала загрузочного сектора определяется значением конфигурационных бит BOOTSZ. См. раздел “Самопрограммирование из сектора начальной загрузки с поддержкой чтения во время записи” для выяснения подробностей. Во избежание несанкционированных изменений таблицы векторов прерываний необходимо выполнить специальную последовательность записи при изменении бита IVSEL:

1. Записать лог. 1 в бит разрешения изменения вектора прерывания (IVCE).
2. В течение четырех машинных циклов записать желаемое значение в IVSEL, при этом записывая лог.0 в IVCE.

Прерывания будут автоматически отключены при выполнении такой последовательности. Прерывания отключаются во время установки IVCE и останутся отключенными до перехода к инструкции следующей за инструкцией записи в IVSEL. Если IVSEL не записан, то прерывания будет находиться в отключенном состоянии 4 такта синхронизации. Состояние бита I в регистре статуса не затрагивается при автоматическом отключении прерываний.

Прим. : Если векторы прерываний помещаются в загрузочный сектор и бит защиты загрузочного сектора BLB02 запрограммирован, то прерывания будут отключены при выполнения программы с секторе прикладной программы. Если векторы прерываний размещены в прикладном секторе и бит защиты BLB12 запрограммирован, то прерывания становятся отключенными при выполнении программы в загрузочном секторе. См. также “ Самопрограммирование из сектора начальной загрузки с поддержкой чтения во время записи ” для более подробного изучения бит защиты.

– IVCE: Разрешение Изменения Вектора Прерывания

В бит IVCE должна быть записана лог. 1, чтобы разрешить изменение бита IVSEL. IVCE сбрасывается аппаратно через четыре машинных цикла после записи лог. 1 в IVSEL. Установка бита IVCE приведет к отключению прерываний, что описано при рассмотрении бита IVSEL выше. Ниже приведен пример кода.

Пример кода на Ассемблере

```
Move_interrupts:
; Разрешение изменения векторов прерываний
ldi r16, (1<<IVCE)

out GICR , r16
; Перемещение векторов в загрузочный сектор флэш-памяти
ldi r16, (1<<IVSEL)

out GICR , r16
ret
```

Пример кода на Си

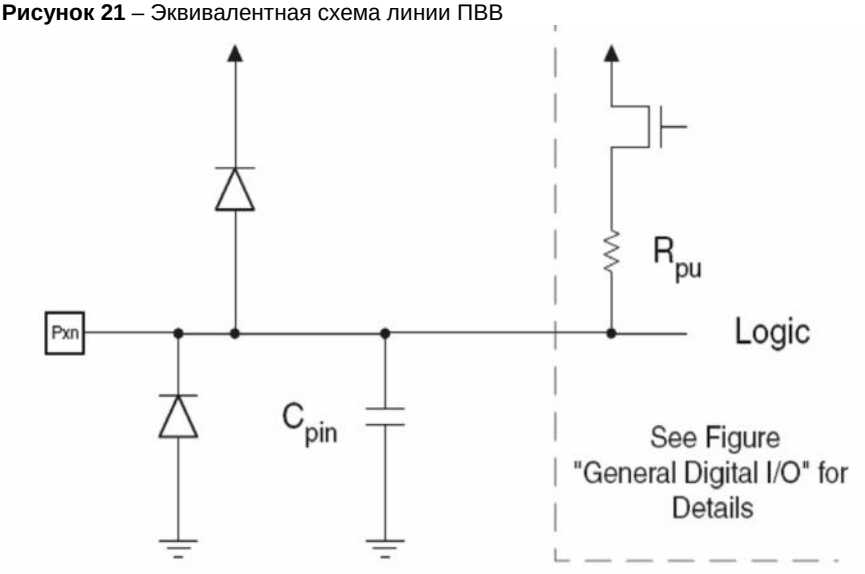
```
void Move_interrupts(void)
{
/* Разрешение изменения векторов прерываний */
GICR = (1<<IVCE);
/* Перемещение векторов в загрузочной сектор флэш-памяти */
GICR = (1<<IVSEL);
}
```

Порты ввода-вывода

Введение

Все порты ввода-вывода (ПВВ) AVR-микроконтроллеров работают по принципу чтение-модификация-запись при использовании их в качестве портов универсального ввода-вывода. Это означает, что изменение направления ввода-вывода одной линии порта командами SBI и CBI будет происходит без ложных изменений направления ввода-вывода других линий порта. Данное распространяется также и на изменение логического уровня (если линия порта настроена на вывод) или на включение/отключение подтягивающих резисторов (если линия настроена на ввод). Каждый выходной буфер имеет симметричную характеристику управления с высоким втекающим и вытекающим выходными токами. Выходной драйвер обладает нагрузочной способностью, которая позволяет непосредственно управлять светодиодными индикаторами. Ко всем линиям портов может быть подключен индивидуальный выборочный подтягивающий к плюсу питания резистор, сопротивление которого не зависит от напряжения питания. На всех линиях ПВВ установлены защитные диоды, которые подключены к VCC и Общему (GND), как показано на рисунке 21. Подробный перечень параметров ПВВ приведен в разделе "Электрические характеристики".

Рисунок 21 – Эквивалентная схема линии ПВВ



Ссылки на регистры и биты регистров в данном разделе даны в общей форме. При этом, символ “x” заменяет наименование ПВВ, а символ “n” заменяет номер разряда ПВВ. Однако при составлении программы необходимо использовать точную форму записи. Например, PORTB3, означающий разряд 3 порта B, в данном документе записывается как PORTxn. Адреса физических регистров ввода-вывода и распределение их разрядов приведены в разделе “Описание регистров портов ввода-вывода".

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

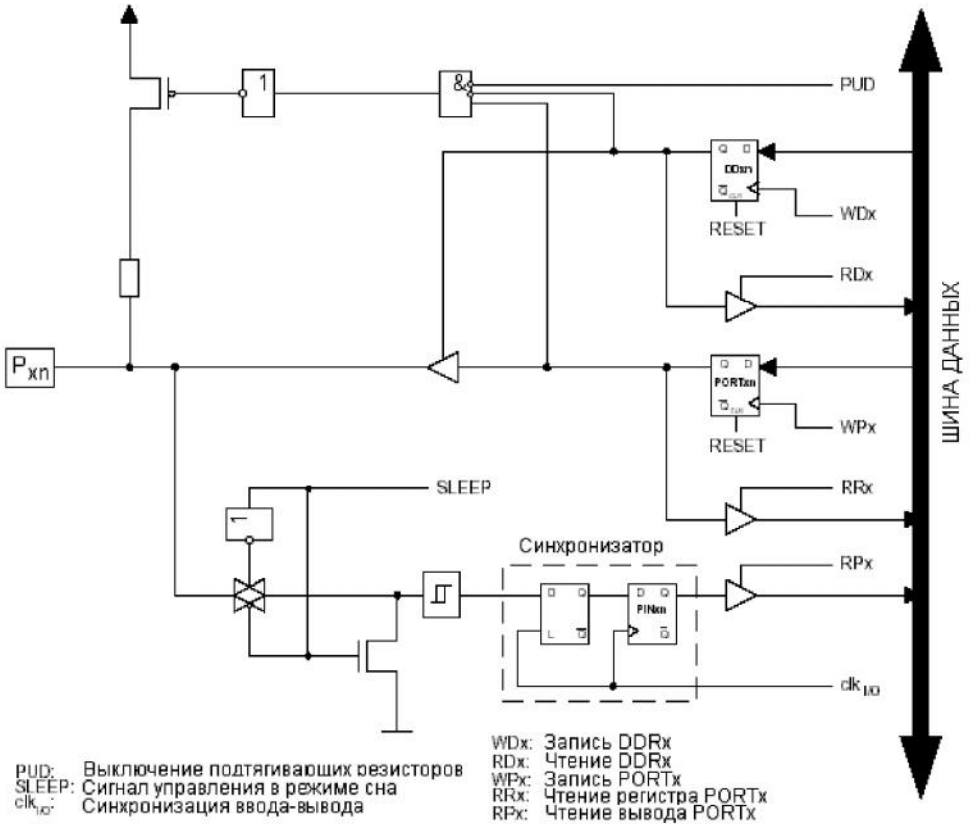
Ниже приведено описание порта ввода-вывода для универсального цифрового ввода-вывода. Большинство выводов портов поддерживают альтернативные функции встроенных периферийных устройств микроконтроллера. Описание альтернативных функций приведено далее в подразделе “Альтернативные функции порта” (см. также описание функций соответствующих периферийных модулей).стр.56

Отметьте, что включение дополнительной функции некоторых из контактов порта не затрагивает использование других контактов в порту для универсального цифрового ввода-вывода.

Порты в качестве универсального цифрового ввода-вывода

Все порты являются двунаправленными портами ввода-вывода с опциональными подтягивающими резисторами. Рисунок 22 иллюстрирует функциональную схему одной линии порта ввода-вывода, обозначенный как Pxn.

Рисунок 22. Организация универсального цифрового ввода-вывода (1)



Прим. 1: Сигналы WPx, WDx, RRx, RPx и RDx являются общими в пределах одного порта. Сигналы clkI/O, SLEEP, и PUD являются общими для всех портов.

Настройка выводов

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

Биты DDxn регистра DDRx определяют направленность линии ввода-вывода. Если DDxn = 1, то Pxn конфигурируется на вывод. Если DDxn=0, то Pxn конфигурируется на ввод.

Если PORTxn = 1 при конфигурации линии порта на ввод, то разрешается подключение подтягивающего резистора. Для выключения данного резистора необходимо записать в PORTxn лог. 0 или настроить линию порта на вывод. Во время сброса все линии портов находятся в третьем (высокоимпедансном) состоянии, даже если не работает синхронизация.

При конфигурации линии порта на вывод, состояние выхода будет определяться значением PORTxn.

Поскольку одновременная запись в регистры DDRx и PORTx невозможна, то при переключении между третьим состоянием ({DDxn, PORTxn} = 0b00) и выводом лог. 1 ({DDxn, PORTxn} = 0b11) должно возникнуть промежуточное состояние или с подключенным подтягивающим резистором ({DDxn, PORTxn} = 0b01) или с выводом лог. 0 ({DDxn, PORTxn} = 0b10). Как правило, переход через состояние с подключением подтягивающего резистора эквивалентно состоянию вывода лог.1, если вывод микроконтроллера связан с высокоимпедансным входом. В противном случае, необходимо установить бит PUD регистра SFIOR для выключения всех подтягивающих резисторов на всех портах

Переключение между вводом с подтягивающими резисторами и выводом низкого уровня связано с аналогичной проблемой. Поэтому, пользователь вынужден использовать или третье состояние ({DDxn, PORTxn} = 0b00) или вывод лог. 1 ({DDxn, PORTxn} = 0b11) в качестве промежуточного шага.

В таблице 20 подытоживается действие управляющих сигналов на состояние вывода.

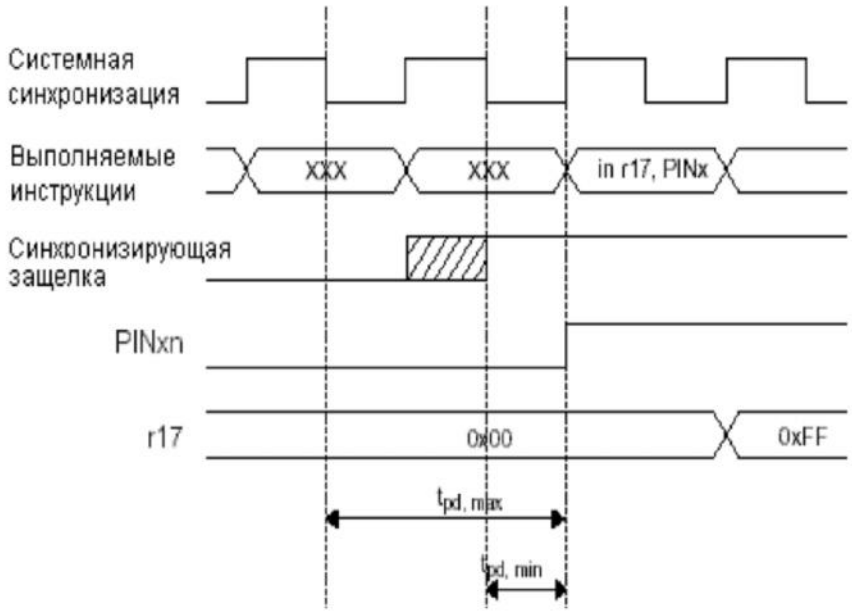
Таблица 20 – Настройка вывода порта

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

Считывание состояние вывода

Независимо от значения бита направления данных DDxn состояние вывода порта может быть опрошено через регистровый бит PINxn. Как показано на рисунке 22 регистровый бит PINxn и предшествующая ему триггерная защелка составляют синхронизатор. Данный подход позволяет избежать метастабильности, если изменение состояния на выводе произошло около фронта внутренней синхронизации. Однако такой подход связан с возникновением задержки. На рисунке 23 представлена временная диаграмма синхронизации во время опроса внешне приложенного к выводу уровня. Длительности минимальной и максимальной задержек на распространение сигнала обозначены как tpd,max и tpd,min, соответственно.

Рисунок 23 – Синхронизация во время опроса приложенного к выводу порта уровня



Рассмотрите период времени, запускающийся вскоре после первого падающего фронта тактового сигнала. Защелка закрывается, когда тактовый сигнал низкий, и не реагирует, когда тактовый сигнал высокий, как обозначаются теневой областью “SYNC LATCH”сигнала. Сигнальное значение фиксируют, когда тактовый сигнал низкий. Это синхронизируется в регистр PINxn в последующем положительном фронте синхроимпульса. Как обозначено этими двумя стрелками tpd, максимальный и tpd, минимальный, единственный сигнальный переход на контакте будет задержан между ½ и 1-½ системными периодами тактового сигнала в зависимости от времени поступления сигнала.

При программном чтении состояния штырька , nop инструкция должна быть включена , как указано на Рисунке 24. Инструкция устанавливает сигнал “SYNC LATCH” (БЛОКИРОВКИ СИНХРОНИЗАЦИИ) на положительном фронте тактового сигнала. В этом случае, задержка tpd синхронизации составляет - 1 период тактового сигнала.

Рисунок 24. Синхронизация при программном чтении значения контакта

В следующих примерах показано как установить на линиях 0 и 1 порта B уровень лог. 1, а на линиях 2 и 3 – лог. 0, а также как настроить линии 4…7 на ввод с подключением подтягивающих резисторов на линиях 6 и 7. Результирующее состояние линий считываются обратно, но, с учетом сказанного выше, включена инструкция nop для обеспечения возможности обратного считывания только что назначенного состояния некоторых выводов.

Пример кода на Ассемблере (1)

```
...
; Разрешаем подтягивание и устанавливаем высокие выходные уровни
; Определяем направления данных линий портов
ldi r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out PORTB,r16
out DDRB,r17
; Вставляем инструкцию nop для синхронизации
nop
; Опрос состояния выводов порта
in r16,PINB
...
```

Пример кода на Си (1)

```
unsigned char i;
...
/* Разрешаем подтягивание и устанавливаем высокие выходные уровни
*/
/* Определяем направления данных линий портов */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Вставляем инструкцию nop для синхронизации */
_NOP();
/* Опрос состояния выводов порта*/
i = PINB;
...
```

Прим.1 В программе на Ассемблере используются два временных регистра для минимизации интервала времени от настройки подтягивающих резисторов на разрядах 0, 1, 6 и 7 до корректной установки бит направления, разрешающих вывод лог. 0 на линиях 2 и 3 и заменяющих высокий уровень на разрядах 0 и 1, образованный за счет подключения подтягивающих резисторов, на высокий уровень сильноточного драйвера.

Разрешение цифрового ввода и режимы сна

Как показано на рисунке 22 входной цифровой сигнал может быть зашунтирован к общему на входе триггера Шмита. Сигнал, обозначенный на рисунке как SLEEP, устанавливается при переводе микроконтроллера в режим выключения (Power-down), экономичный режим, дежурный режим и расширенный дежурный режим. Это позволяет избежать повышения потребляемого тока в случае, если некоторые входные сигналы окажутся в плавающем состоянии или уровень входного аналогового сигнала будет близок к VCC/2.

Сигнал SLEEP игнорируется по входам внешних прерываний. Если запросы на внешнее прерывание отключены, то SLEEP действует и на эти выводы. SLEEP также игнорируется на некоторых других входах при выполнении их альтернативных функций (см. “Альтернативные функции порта ”).

Если на выводе внешнего асинхронного прерывания, настроенный на прерывание по нарастающему фронту, падающему фронту или на любое изменение, присутствует уровень лог. 1 и при этом внешнее прерывание не разрешено, то соответствующий флаг внешнего прерывания будет установлен при выходе из выше упомянутых режимов сна, т.к. функция шунтирования на входе в режимах сна приводит возникновению логических изменений.

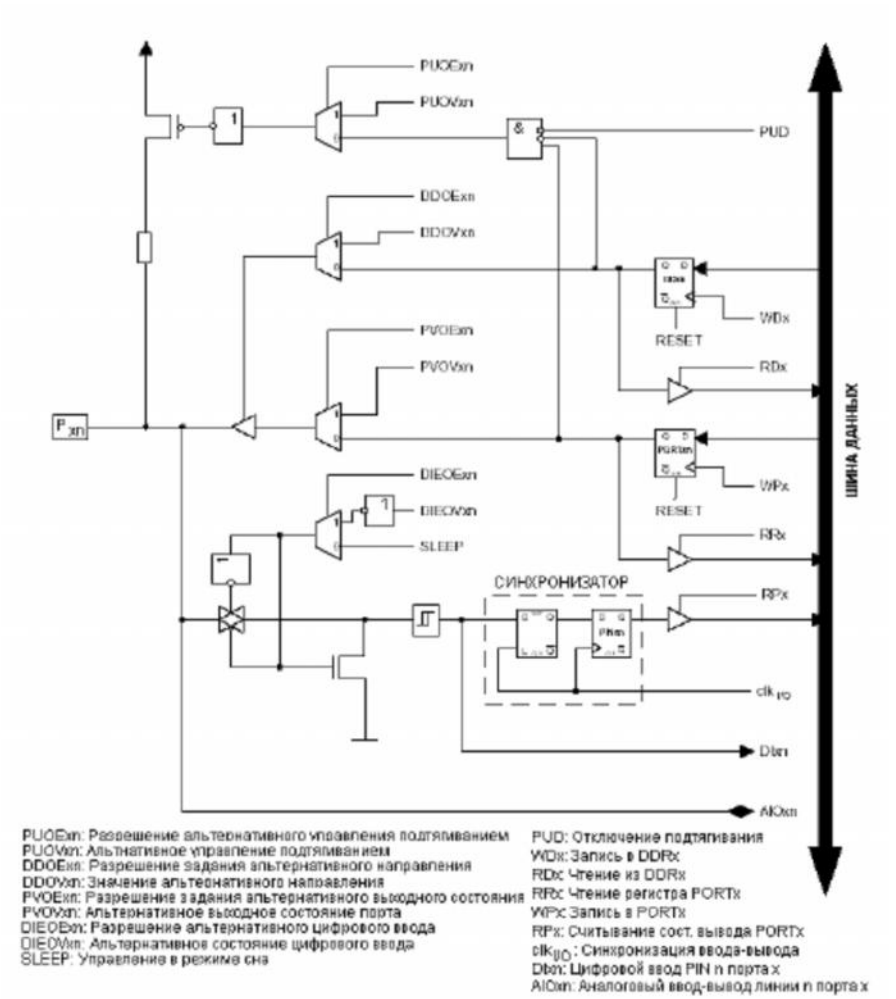
Неподключенные выводы

Если несколько выводов остаются неиспользованными, то рекомендуется гарантировать на них присутствие определенного логического уровня. Не смотря на то, что большинство цифровых входов отключены в режимах глубокого сна, как описано выше, необходимо избежать наличия плавающих входов во избежание повышенного потребления тока во всех других режимах работы микроконтроллера, где цифровой ввод разрешен (Сброс, Активный режим и режим холостого).

Самым простым методом гарантирования присутствия определенного уровня на неиспользуемом выводе является разрешение подключения внутреннего подтягивающего резистора. Однако в этом случае в режиме сброса подтягивающие резисторы будут отключены.
Если требуется малое потребление и в режиме сброса, то необходимо устанавливать внешний подтягивающий резистор к плюсу или к минусу питания. Подключение выводов непосредственно к VCC или GND не рекомендуется, т.к. может возникнуть опасный ток при случайной конфигурации такого вывода на вывод данных.

Альтернативные функции порта

Большинство выводов поддерживают альтернативные функции в дополнение к универсальному цифровому вводу-выводу. На рисунке 25 показано как управляющие сигналы, представленные на упрощенном рисунке 22, могут быть отключены альтернативными функциями. Сигналы отключения могут присутствовать не на всех выводах, поэтому, данный рисунок необходимо использовать как общее описание, применимое ко всем выводам портов семейства AVR- микроконтроллеров.
Рисунок 25. Альтернативные Функции Порта (1)



Прим. 1: Сигналы WPx, WDx, RLx, RPx и RDx являются общими в пределах одного порта. Сигналы clkI/O, SLEEP, и PUD являются общими для всех портов. Все остальные сигналы индивидуальны для каждого вывода.

В таблице 21 подытожены функции отключающих сигналов для активизации альтернативных функций. Указатели на выводы и порты с рисунка 33 не показаны в итоговых таблицах. Отключающие сигналы генерируются внутренне в модулях, поддерживающих альтернативные функции.

Таблица 21 – Общее описание отключающих сигналов для активизации альтернативных функций

Наименование сигнала	Полное наименование	Описание
PUOE	Разрешение альтернативного управления подтягиванием	Если данный сигнал установлен, то подключение подтягивающего резистора определяется значением сигнала PUOV. Если данный сигнал сброшен, то подтягивающий резистор подключается, если {DDxn, PORTxn, PUD} = 0b010.
PUOV	Альтернативное управление подтягиванием	Если PUOE установлен, то подтягивающий резистор подключается/отключается, если PUOV установлен/сброшен независимо от состояния регистровых бит DDxn, PORTxn и PUD.
DDOE	Разрешение задания альтернативного направления	Если этот сигнал установлен, то разрешение работы выходного драйвера определяется значением сигнала DDOV. Если этот сигнал сброшен, то работа выходного драйвера разрешается регистровым битом DDxn.
DDOV	Значение альтернативного направления	Если DDOE установлен, то работа выходного драйвера разрешается/запрещается, когда DDOV устанавливается/сбрасывается независимо от состояния регистрового бита DDxn.
PVOE	Разрешение задания альтернативного выходного состояния порта	Если данный сигнал установлен и разрешена работа выходного драйвера, то состояние на выходе порта определяется сигналом PVOV. Если PVOE сброшен и разрешена работа выходного драйвера, то состояние на выходе порта определяется регистровым битом PORTxn.
PVOV	Альтернативное выходное состояние порта	Если PVOE установлен, то выход порта принимает состояние PVOV независимо от установки регистрового бита PORTxn.
DIEOE	Разрешение альтернативного цифрового ввода	Если данный бит установлен, то функция разрешения цифрового передается сигналу DIEOV. Если данный сигнал сброшен, то разрешение цифрового ввода определяется состоянием микроконтроллера (нормальный режим, режимы сна).
DIEOV	Альтернативное состояние цифрового ввода	Если DIEOE установлен, то цифровой ввод разрешен/запрещен, если DIEOV установлен/сброшен независимо от состояния микроконтроллера (нормальный режим, режимы сна).
DI	Цифровой ввод	Сигнал цифрового ввода для альтернативных функций. На рисунке сигнал подключен к выходу триггера Шмита перед синхронизатором. Если цифровой ввод используется как источник синхронизации, то модуль с альтернативной функцией будет использовать свой собственный синхронизатор.
AIO	Аналоговый ввод-вывод	Сигнал аналогового ввода/вывода к_модулю/из_модуля с альтернативной функцией. Сигнал подключается непосредственно к контактной площадке и может использоваться двунаправлено.

В следующих подразделах коротко описываются альтернативные функции для каждого порта и связь отключающих сигналов с альтернативными функциями выводов.

Регистр специальных функций ввода-вывода – SFIOR

Разряд	7	6	5	4	3	2	1	0	
	TSM	–	–	–	ACME	PUD	PSR0	PSR321	SFIOR
Чтение/Запись	Чт./Зп.	Чт.	Чт.	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Начальное знач.	0	0	0	0	0	0	0	0	

– PUD: Отключение всех подтягивающих резисторов

Если в данный разряд записать лог. 1, то подтягивающие резисторы на всех портах будет отключены, даже если регистры DDxn и PORTxn настроены на их подключение ({DDxn, PORTxn} =0b01). См. “Настройка выводов” для детального изучения данной функции.

Альтернативные функции порта B

Выводы порта B с альтернативными функциями показаны в таблице 22.

Таблица 22 – Альтернативные функции порта B

Вывод порта	Альтернативная функция
PB7	XTAL2 (выход для подключения резонатора контакт 2) TOSC2 (Контакт резонатора таймера 2)
PB6	XTAL1 (выход для подключения резонатора контакт 1) TOSC1(Контакт резонатора таймера 1)
PB5	SCK (Синхронизация последовательной связи шины SPI)
PB4	MISO (Ввод для ведущей/вывод для подчиненной шины SPI)
PB3	MOSI (Вывод для ведущей/ввод для подчиненной шины SPI)
PB2	SS (вход выбора подчиненного режима интерфейса SPI) OC1B (выход B компаратора и ШИМ таймера-счетчика 1)
PB1	OC1A (выход A компаратора и ШИМ таймера-счетчика 1)
PB0	ICP1(вход триггера захвата фронта таймера-счетчика 1)

Альтернативная конфигурация контакта следующая:

• XTAL2/TOSC2 – Port B, Bit 7

XTAL2: Pin 2 для генератора синхросигналов микросхемы. Используемый в качестве тактового вывода для кварцевого генератора или низкочастотного кварцевого генератора. Когда он используется в качестве тактового вывода, контакт не может использоваться в качестве контакта ввода-вывода.
TOSC2: Pin 2 резонатора Таймера2. Используемый, только если внутренний калиброванный Осциллятор RC выбирается как источник часов микросхемы, и асинхронный таймер включается корректной установкой в ASSR.
Когда бит AS2 ASSR устанавливается, чтобы включить асинхронную синхронизацию Timer/Counter2, pin PB7 разъединяется от порта, и становится выводом инвертирования усилителя Осциллятора. В этом режиме кварцевый генератор соединяется с этим контактом, и контакт не может использоваться в качестве контакта ввода-вывода.
Если PB7 будет использоваться в качестве тактового вывода, то DDB7, PORTB7 и PINB7 будут все читаться как 0.

• XTAL1/TOSC1 – Port B, Bit 6

XTAL1: : Pin 1 для генератора синхросигналов микросхемы. Используемый в качестве тактового вывода для кварцевого генератора или низкочастотного кварцевого генератора. Когда он используется в качестве тактового вывода, контакт не может использоваться в качестве контакта ввода-вывода.
TOSC1: Pin 1 резонатора Таймера2. Используемый, только если внутренний калиброванный Осциллятор RC выбирается как источник часов микросхемы, и асинхронный таймер включается корректной установкой в ASSR.
Когда бит AS2 ASSR устанавливается, чтобы включить асинхронную синхронизацию Timer/Counter2, pin PB6 разъединяется от порта, и становится

выводом инвертирования усилителя Осциллятора. В этом режиме кварцевый генератор соединяется с этим контактом, и контакт не может использоваться в качестве контакта ввода-вывода.
Если PB6 будет использоваться в качестве тактового вывода, то DDB6, PORTB6 и PINB6 будут все читаться как 0.

• SCK – Port B, Bit 5
SCK – выход синхронизации в режиме ведущего, вход синхронизации в режиме подчиненного интерфейса SPI. Если работа SPI разрешена как подчиненного, то данный вывод настраивается как вход независимо от состояния DDB5. Если работа SPI разрешена как ведущего, то направление передачи данных управляется DDB5. Если вывод принудительно настроен на ввод, то управление подтягивающими резисторами осуществляется битом PORTB5.

• MISO – Port B, Bit 4
MISO – ввод данных в режиме ведущего, вывод данных в режиме подчиненного интерфейса SPI. Если разрешена работа SPI как ведущего (мастера), то данный вывод настраивается на ввод независимо от состояния DDB4. Если работа SPI разрешена как подчиненного, то направление передачи данных задается DDB4. Если вывод принудительно настраивается на ввод, то подключение подтягивающего резистора останется под управлением бита PORTB4.

• MOSI/OC2 – Port B, Bit 3
MOSI – вывод данных в режиме ведущего, ввод данных в режиме подчиненного интерфейса SPI. Если работа SPI разрешена как подчиненного, то данный вывод настраивается на ввод независимо от значения DDB3. Если работа SPI разрешена как ведущего (мастера), направление передачи данных определяется DDB3. Если вывод принудительно настраивается как вход, то подключение подтягивающего резистора останется под управлением PORTB3.
OC2 – выход компаратора таймера-счетчика 2. Для выполнения данной функции вывод PB7 конфигурируется как выход (DDB3 = 1). Вывод OC2 также выполняет функцию выхода, когда таймер переводится в режим ШИМ.

• SS/OC1B – Port B, Bit 2
SS - вход выбора подчиненного порта. Если работа SPI разрешена как подчиненного, то данный вывод настраивается на ввод независимо от установки DDB2. Работа SPI как подчиненного активизируется, если подать низкий уровень на этот вход. Если работа SPI разрешена как ведущего, то направление передачи данных на этом выводе задается DDB2. Если вывод принудительно настроить как вход, то подключение подтягивающего резистора управляется битом PORTB2.

• OC1A – Port B, Bit 1
OC1A – выход компаратора A таймера-счетчика 1. Для выполнения данной функции вывод PB1 настраивается как выход (DDB1 = 1). Вывод OC1A также выполняет функцию выхода, когда таймер переведен в режим ШИМ.

• ICP1 – Port B, Bit 0
ICP1 – вход захвата фронта таймера-счетчика 1.

В таблицах 23 и 24 показаны значения отключающих сигналов (см. рис. 25) в различных альтернативных функциях порта B. SPI MSTR INPUT (вход ведущего SPI) и SPI SLAVE OUTPUT (выход подчиненного SPI) составляют сигнал MISO, а сигнал MOSI разделен на SPI MSTR OUTPUT (выход ведущего SPI) и SPI SLAVE INPUT (вход подчиненного SPI).

Таблица 23. Отключающие сигналы для альтернативных функций на PB7..PB4

Signal Name	PB7/XTAL2/ TOSC2[1][2]	PB6/XTAL1/ TOSC1[1]	PB5/SCK	PB4/MISO
PUOE	$\overline{EXT}$ • ($\overline{INTRC}$ + AS2)	$\overline{INTRC}$ + AS2	SPE • $\overline{MSTR}$	SPE • MSTR
PUO	0	0	PORTB5 • $\overline{PUD}$	PORTB4 • $\overline{PUD}$
DDOE	$\overline{EXT}$ • ($\overline{INTRC}$ + AS2)	$\overline{INTRC}$ + AS2	SPE • $\overline{MSTR}$	SPE • MSTR
DDOV	0	0	0	0
PVOE	0	0	SPE • MSTR	SPE • $\overline{MSTR}$
PVOV	0	0	SCK OUTPUT	SPI SLAVE OUTPUT
DIEOE	$\overline{EXT}$ • ($\overline{INTRC}$ + AS2)	$\overline{INTRC}$ + AS2	0	0
DIEOV	0	0	0	0
DI	–	–	SCK INPUT	SPI MSTR INPUT
AIO	Oscillator Output	Oscillator/Clock Input	–	–

Примечания: 1. INTRC означает, что внутренний генератор RC выбирается (битом CKSEL).

3. EXT означает, что внешний генератор RC или внешний таймер выбираются (битом CKSEL).

4.

Таблица 24. Отключающие сигналы для альтернативных функций на PB3..PB0

Signal Name	PB3/MOSI/OC2	PB2/$\overline{SS}$/OC1B	PB1/OC1A	PB0/ICP1
PUOE	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$	0	0
PUO	PORTB3 • $\overline{PUD}$	PORTB2 • $\overline{PUD}$	0	0
DDOE	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$	0	0
DDOV	0	0	0	0
PVOE	SPE • MSTR + OC2 ENABLE	OC1B ENABLE	OC1A ENABLE	0
PVOV	SPI MSTR OUTPUT + OC2	OC1B	OC1A	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	SPI SLAVE INPUT	SPI $\overline{SS}$	–	ICP1 INPUT
AIO	–	–	–	–

Альтернативные функции порта C

Таблица 25 – Альтернативные функции выводов порта C

вывод порта	Альтернативная функция
PC6	RESET(Контакт сброса)
PC5	ADC5(Входной Канал ADC 5) SCL(линия синхронизации Двухпроводной последовательной шины)
PC4	ADC4 (Входной Канал ADC 4) SDA (Ввод / Вывод данных Двухпроводной последовательной шины)
PC3	ADC3 (Входной Канал ADC 3)
PC2	ADC2 (Входной Канал ADC 2)
PC1	ADC1 (Входной Канал ADC 1)
PC0	ADC0 (Входной Канал ADC 0)

Альтернативная конфигурации контакта следующая:

• RESET – Port C, Bit 6

СБРОС, контакт Сброса: Когда бит RSTDISBL программируется, этот контакт функционирует как нормальный контакт ввода-вывода, и чип должен будет расчитывать на сброс по включению питания и снижению напряжения как на источники сброса. Когда бит RSTDISBL непрограммируется, схема сброса соединяется с контактом, и контакт не может использоваться в качестве контакта ввода-вывода.

Если PC6 будет использоваться в качестве контакта сброса, то DDC6, PORTC6 и PINC6 будут всегда читаться как 0.

• SCL/ADC5 – Port C, Bit 5

SCL – синхронизация двухпроводного последовательного интерфейса. Если установлен бит TWEN в регистре TWCR, то разрешается работа двухпроводного последовательного интерфейса, вывод PC5 отключается от порта и становится входом/выходом синхронизации последовательной
связи двухпроводного последовательного интерфейса. В этом режиме на входе активизируется помехоподавляющий фильтр, который не реагирует на входные импульсы длительностью менее 50 нс, а передача организована драйвером с открытым стоком и ограниченной скоростью изменения сигнала.

PC5 может также использоваться в качестве входного Канала ADC 5. Отметьте, что входной канал ADC 5 использует цифровое питание.

• SDA/ADC4 – Port C, Bit 4

SDA – ввод-вывод данных двухпроводного последовательного интерфейса . После установки бита TWEN в регистре TWCR разрешается работа двухпроводного последовательного интерфейса, вывод PC4 отключается от порта и становится линией ввода-вывода последовательных данных двухпроводного последовательного интерфейса. В этом режиме на визируется помехоподавляющий фильтр, который не реагирует на входные импульсы длительностью менее 50 нс, а передача организована драйвером с открытым стоком и ограниченной скоростью изменения сигнала.

PC4 может также использоваться в качестве входного Канала ADC 4. Отметьте, что входной канал ADC 4 использует цифровое питание.

• ADC3 – Port C, Bit 3

PC3 может также использоваться в качестве входного Канала ADC 3. Отметьте, что входной канал ADC 3 использует аналоговое питание.

• ADC2 – Port C, Bit 2

PC2 может также использоваться в качестве входного Канала ADC 2. Отметьте, что входной канал ADC 2 использует аналоговое питание.

• ADC1 – Port C, Bit 1

PC1 может также использоваться в качестве входного Канала ADC 1. Отметьте, что входной канал ADC 1 использует аналоговое питание.

• ADC0 – Port C, Bit 0
PC0 может также использоваться в качестве входного Канала ADC 0. Отметьте, что входной канал ADC 0 использует аналоговое питание.

В таблицах 26 и 27 представлена связь альтернативных функций порта C и отключающих сигналов, представленных на рисунке 25.

Таблица 26 – Отключающие сигналы для разрешения альтернативных функций на PC6..PC4

Signal Name	PC6/$\overline{\text{RESET}}$	PC5/SCL/ADC5	PC4/SDA/ADC4
PUOE	$\overline{\text{RSTDISBL}}$	TWEN	TWEN
PUOV	1	PORTC5 • $\overline{\text{PUD}}$	PORTC4 • $\overline{\text{PUD}}$
DDOE	$\overline{\text{RSTDISBL}}$	TWEN	TWEN
DDOV	0	SCL_OUT	SDA_OUT
PVOE	0	TWEN	TWEN
PVOV	0	0	0
DIEOE	$\overline{\text{RSTDISBL}}$	0	0
DIEOV	0	0	0
DI	–	–	–
AIO	RESET INPUT	ADC5 INPUT / SCL INPUT	ADC4 INPUT / SDA INPUT

Таблица 27 – Отключающие сигналы для разрешения альтернативных функций на PC3..PC0(1)

Signal Name	PC3/ADC3	PC2/ADC2	PC1/ADC1	PC0/ADC0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	–	–	–	–
AIO	ADC3 INPUT	ADC2 INPUT	ADC1 INPUT	ADC0 INPUT

Прим. 1: После разрешения работы TWI активизируется схема управления скоростью изменения выходных сигналов на выводах PC4 и PC5. Данная функция не учтена в таблице. Кроме того, помехоподавляющие фильтры подключены между выходами AIO и цифровой логикой модуля TWI.

Альтернативные Функции Порта D

Альтернативные функции выводов порта D показаны Таблице 28.

Таблица 28 – Альтернативные функции выводов порта D

Port Pin	Alternate Function
PD7	AIN1 (Analog Comparator Negative Input)
PD6	AIN0 (Analog Comparator Positive Input)
PD5	T1 (Timer/Counter 1 External Counter Input)
PD4	XCK (USART External Clock Input/Output) T0 (Timer/Counter 0 External Counter Input)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

Альтернативная конфигурации контакта следующая:

• AIN1 – Port D, Bit 7
AIN1 – инвертирующий вход аналогового компаратора. Сконфигурируйте штырек порта как вход с выключенным внутренним подтягивающим напряжением, чтобы избежать цифровой функции порта и не создавать помехи функционированию Аналогового Компаратора.

• AIN0 – Port D, Bit 6
AIN0 – неинвертирующий вход аналогового компаратора. Сконфигурируйте штырек порта как вход с выключенным внутренним подтягивающим напряжением, чтобы избежать цифровой функции порта и не создавать помехи функционированию Аналогового Компаратора.

• T1 – Port D, Bit 5
T1 – счетный вход таймера-счетчика 1.

• XCK/T0 – Port D, Bit 4
XCK – внешняя синхронизация УСАПП1.
T0 – счетный вход таймера-счетчика

• INT1 – Port D, Bit 3
INT1 – источник внешнего прерывания 1. Вывод PD3 может использоваться как источник внешнего прерывания микроконтроллера.

• INT0 – Port D, Bit 2
INT0 – источник внешнего прерывания 0. Вывод PD2 может использоваться как источник внешнего прерывания микроконтроллера.

• TXD – Port D, Bit 1
TXD – передача данных (вывод данных для УСАПП1). Если работа передатчика УСАПП1 разрешена, то данный вывод настраивается как выход независимо от значения DDD1.

• RXD – Port D, Bit 0
RXD1 – прием данных (ввод данных для УСАПП1). Если работа приемника УСАПП1 разрешена, то данный вывод настраивается на ввод независимо от значения DDD0. После перевода УСАППом данного вывода на вход, управление подтягивающим резистором осуществляется битом PORTD0.

Таблица 29 и Таблица 30 показывают связь между альтернативными функциями выводов порта D и отключающими сигналами, показанными на рисунке 25.

Таблица 29. Отключающие сигналы для разрешения альтернативных функций на PD7..PD4

Signal Name	PD7/AIN1	PD6/AIN0	PD5/T1	PD4/XCK/T0
PUOE	0	0	0	0
PUO	0	0	0	0
OOE	0	0	0	0
OO	0	0	0	0
PVOE	0	0	0	UMSEL
PVO	0	0	0	XCK OUTPUT
DIEOE	0	0	0	0
DIEO	0	0	0	0
DI	–	–	T1 INPUT	XCK INPUT / T0 INPUT
AIO	AIN1 INPUT	AIN0 INPUT	–	–

Таблица 30. Отключающие сигналы для разрешения альтернативных функций на PD3..PD0

Signal Name	PD3/INT1	PD2/INT0	PD1/TXD	PD0/RXD
PUOE	0	0	TXEN	RXEN
PUO	0	0	0	PORTD0 • $\overline{PUD}$
OOE	0	0	TXEN	RXEN
OO	0	0	1	0
PVOE	0	0	TXEN	0
PVO	0	0	TXD	0
DIEOE	INT1 ENABLE	INT0 ENABLE	0	0
DIEO	1	1	0	0
DI	INT1 INPUT	INT0 INPUT	–	RXD
AIO	–	–	–	–

Описание регистров портов ввода-вывода

Регистр данных порта B – PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Регистр направления данных порта B – DDRB

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Адрес входов порта B – PINB

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Регистр данных порта C – PORTC

Bit	7	6	5	4	3	2	1	0	
	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Регистр направления данных порта C – DDRC

Bit	7	6	5	4	3	2	1	0	
	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Адрес входов порта C – PINC

Bit	7	6	5	4	3	2	1	0	
	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Регистр данных порта D – PORTD

Bit	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Регистр направления данных порта D – DDRD

Bit	7	6	5	4	3	2	1	0	
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Адрес входов порта D – PIND

Bit	7	6	5	4	3	2	1	0	
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Внешние прерывания

Внешние прерывания осуществляются через выводы INT0 и INT1. Обратите внимание, что после разрешения внешние прерывания будут генерироваться, даже если линии INT01 настроены как выходы. Данная особенность может использоваться для программной генерации прерывания. Внешние прерывания могут генерироваться по подающему или нарастающему фронту, а также по низкому лог. уровню. Одна из этих установок задается в регистре команд MCU – MCUCR.
Если внешнее прерывание разрешено и настроено на срабатывание при низком уровне, то прерывание будет инициироваться постоянно пока на выводе будет оставаться низкий уровень. Обратите внимание, что для распознавания падающего или нарастающего фронтов на INT0:1 необходимо наличие синхронизации ввода-вывода, описанной в разделе “Источники синхронизации и их распределение".
Прерывания по низкому уровню и фронтам на INT0:1 определяются асинхронно. Это означает, что данные прерывания могут использоваться для пробуждения микроконтроллера из режимов глубокого сна.
Синхронизация ввода-вывода останавливается во всех режимах сна за исключением режима холостого хода (Idle).

Обратите внимание, что при использовании прерывания по уровню для пробуждения микроконтроллера из режима выключения (Power-down), только после удержания изменившегося уровня в течение определенного времени генерируется прерывание. Это делает микроконтроллер менее чувствительным к шумам. Оценка изменения состояния уровня выполняется по двум его выборкам с интервалом равным периоду сторожевого таймера, который равен 1 мкс (номинальное значение) при 5.0В и 25°C. Частота сторожевого таймера зависит от напряжения (см.“Электрические характеристики”). Пробуждение микроконтроллера наступает, если на входе присутствует требуемый уровень в процессе выборок или если он удерживается до окончания задержки при запуске синхронизации (возникает при выходе из режимов сна). Время запуска определяется конфигурационными битами SUT (см. “ Источники синхронизации и их распределение”). Если дважды выполнена выборка уровня с синхронизацией сторожевым таймером, но по истечении времени запуска этот уровень исчез, то пробуждение микроконтроллера наступит, но прерывание не будет сгенерировано. Для того чтобы активизировать прерывание по уровню необходимо, чтобы этот уровень удерживался в течение достаточного для пробуждения микроконтроллера времени.

Управляющий Регистр MCU MCUCR

Регистр управления MCU содержит биты управления для управления смыслом прерывания и общих функций MCU.

Bit	7	6	5	4	3	2	1	0	
	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Бит 3, 2 – ISC11, ISC10: Управление Смыслом Прерывания 1 бит 1 и Бит 0

Внешнее Прерывание 1 активируется внешним контактом INT1, если SREG I-bit и соответствующая маска прерывания в GICR устанавливаются. Уровень и фронт на внешнем контакте INT1, которые активируют прерывание, определяются в Таблице 31. Значение на контакте INT1 выбирается прежде, чем обнаружить фронт. Если прерывание по фронту или уровню выбрано то, импульсы, которые длятся дольше, чем один период синхронизации генерирует прерывание. Более короткие импульсы, не гарантируют, прерывания. Если низкоуровневое прерывание выбирается, низкий уровень должен сохраниться, до завершения выполняющейся инструкции генерирования прерывания.

Таблица 31. Управление Смыслом прерывания 1

ISC11	ISC10	Описание
0	0	Низкий уровень INT1 генерирует запрос прерывания.
0	1	Любое логическое изменение на INT1 генерирует запрос прерывания.
1	0	Падающий фронт INT1 генерирует запрос прерывания.
1	1	Возрастающий фронт INT1 генерирует запрос прерывания.

• Бит 1, 0 – ISC01, ISC00: Управление Смыслом Прерывания 0 бит 1 и Бит 0
Внешнее Прерывание 0 активируется внешним контактом INT0, если I-флаг SREG и соответствующая маска прерывания устанавливаются.
Уровень и фронт на внешнем контакте INT0, которые активируют прерывание, определяются в Таблице 32. Значение на контакте INT0 выбирается прежде, чем обнаружить фронт. Если прерывание по фронту или уровню выбрано то, импульсы, которые длятся дольше, чем один период синхронизации генерирует прерывание. Более короткие импульсы, не гарантируют, прерывания. Если низкоуровневое прерывание выбирается, низкий уровень должен сохраниться, до завершения выполняющейся инструкции генерирования прерывания.
Таблица 32. Управление Смыслом прерывания 0

ISC01	ISC00	Описание
0	0	Низкий уровень INT1 генерирует запрос прерывания.
0	1	Любое логическое изменение на INT1 генерирует запрос прерывания.
1	0	Падающий фронт INT1 генерирует запрос прерывания.
1	1	Возрастающий фронт INT1 генерирует запрос прерывания.

Общий Регистр Управления Прерыванием – GICR

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	–	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Бит 7 – INT1: Внешний Запрос Прерывания 1 Включение
Когда бит INT1 устанавливается (1), и I-bit в Регистре Состояния (SREG) устанавливается (1), внешнее прерывание контакта включается. Смысл прерывания определяется битами1/0 (ISC11 и ISC10) в общем Управляющем Регистре MCU (MCUCR), определяют активизировано внешнее прерывание в растущем или падающем фронте или по изменению уровня на штырьке INT1 . Действие на контакте вызовет запрос прерывания, даже если INT1 будет сконфигурирован как вывод. Соответствующее прерывание внешнего запроса прерывания 1 выполняется от вектора прерывания INT1.

• Бит 6 – INT0: Внешний Запрос Прерывания 0 Включение
Когда бит INT1 устанавливается (1), и I-bit в Регистре Состояния (SREG) устанавливается (1), внешнее прерывание контакта включается. Смысл прерывания определяется битами1/0 (ISC01 и ISC00) в общем Управляющем Регистре MCU (MCUCR), определяют активизировано внешнее прерывание в растущем или падающем фронте или по изменению уровня на штырьке INT0 . Действие на контакте вызовет запрос прерывания, даже если INT0 будет сконфигурирован как вывод. Соответствующее прерывание внешнего запроса прерывания 0 выполняется от вектора прерывания INT0.

Общий Регистр Флагов Прерывания – GIFR

Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	–	–	–	–	–	–	GIFR
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

• Бит 7 – INTF1: Внешний Флаг Прерывания 1
Когда событие на контакте INT1 инициировало запрос прерывания, INTF1 принимает единичное состояние.
Если I-bit в SREG и бит INT1 в GICR будут установлены (1), то MCU перейдет к соответствующему вектору прерывания. Флаг очищается, когда подпрограмма прерывания выполняется. Альтернативно, флаг может быть очищен при записи логического 1 в соответствующий бит.
Этот флаг всегда очищается, когда INT1 конфигурируется на прерывание по уровню.

• Бит 6 – INTF0: Внешний Флаг Прерывания 0
Когда событие на контакте INT0 инициировало запрос прерывания, INTF0 принимает единичное состояние.
Если I-bit в SREG и бит INT0 в GICR будут установлены (1), то MCU перейдет к соответствующему вектору прерывания. Флаг очищается, когда подпрограмма прерывания выполняется. Альтернативно, флаг может быть очищен при записи логического 1 в соответствующий бит.
Этот флаг всегда очищается, когда INT0 конфигурируется на прерывание по уровню.

8-разрядный таймер-счетчик 0

Таймер-счетчик0- универсальный, одноканальный, 8- битовый модуль Таймера-счетчика0.

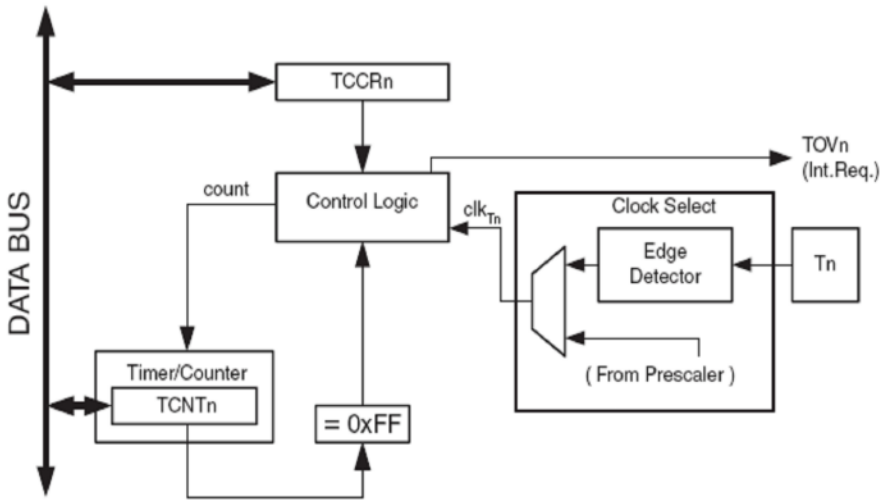
Основные характеристики:

- **Одноканальный счетчик**
- **Генератор частоты**
- **Счетчик внешнего события**
- **10-разрядный делитель тактовой частоты**

Краткий обзор

Укрупненная функциональная схема 8-разр. таймера-счетчика представлена на рис. 26. Для уточнения расположения выводов см. "Расположение выводов"стр.2. Связи с регистрами, к которым осуществляет доступ ЦПУ, в т.ч. биты ввода-вывода и линии ввода-вывода показаны полужирной линией. Специфические для данного устройства регистры, расположение и назначение его бит приведены в "Описание регистров 8-разр. таймера-счетчика 0"стр.72.

Рисунок 26. 8-разрядный Таймер/Счетчик Блок-схема



Регистры

Регистр таймера-счетчика (TCNT0) - 8-разр. регистр. Сигналы запроса на прерывание представлены как флаги прерываний таймера в регистре TIFR. Все прерывания индивидуально маскируются с помощью регистра маски прерываний таймеров (TIMSK). Регистры TIFR и TIMSK не представлены на функциональной схеме, т.к. они совместно используются с другими таймерами микроконтроллера.

Таймер-счетчик может тактироваться через предделитель внутренне или асинхронно через внешний вывод TOSC0. Блок синхронизации осуществляет выбор, какой тактовый источник используется для инкрементирования (декрементирования) состояния таймера-счетчика. Если источник тактирования не задан, то таймер-счетчик находится в неактивном состоянии. Выход логики выбора синхронизации обозначен как синхронизация таймера (clkT0).

Определения

Многие регистры, и разрядные ссылки в этом документе пишутся в общей форме. В более общем случае “n” заменяет число Таймера/Счетчика, в этом случае 0. Однако, при использовании регистра или бита в программе, должна использоваться точная форма записи - TCNT0 для того, чтобы получить значение Таймера/Счетчика0.

Некоторые определения и их сокращенные наименования, которые интенсивно используются в этом разделе, представлены в таблице33

Таблица 33. Определения

НП (нижний предел)	Счетчик достигает нулевого значения (0x00)
МАКС (максимальное значение)	Счетчик достигает максимального значения 0xFF (десятич. 255)

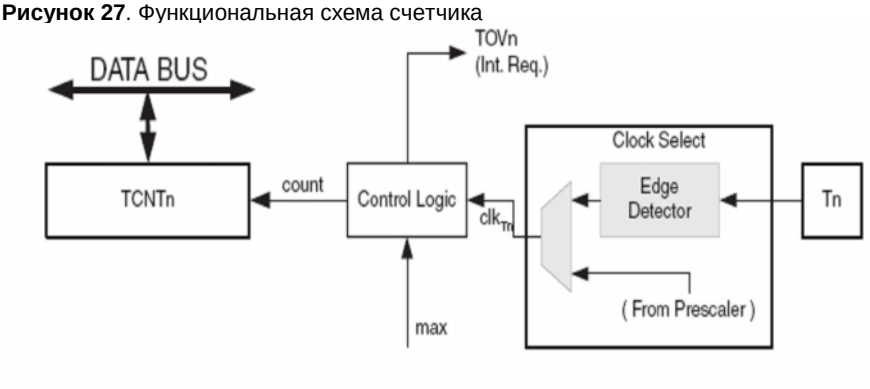
Тактовые источники таймера-счетчика 0

Таймер/Счетчик может тактироваться внутренне синхронно или внешне асинхронно. Источник тактирования выбирается логикой выбора синхронизации, которой управляет выбор (CS02:0) биты, расположенные в Регистре команд Таймера/Счетчика (TCCR0). Для получения дополнительной информации об источниках синхронизации и делителе частоты, см. “Таймер/Счетчик 0 и Делители частоты Таймера/Счетчика1 ” на странице 74.

Блок счетчика

Основу 8-разр. таймера-счетчика 0 составляет программируемый однонаправленный счетчик. Рисунок 27 показывает функциональную схему счетчика и окружающих его элементов.

Рисунок 27. Функциональная схема счетчика



Описание сигналов (внутренние сигналы):
coun - Инкрементирует TCNT0 на 1.
clkT0 - Синхронизация таймера-счетчика.
max - Сигнализирует что, TCNT0 достиг максимального значения.

Счетчик инкрементируется на каждом такте синхронизации (clkT0). Тактовый сигнал clkT0 может быть внутренним или внешним, а его частота выбирается с помощью бит выбора частоты синхронизации (CS02:0). Если источник синхронизации не задан (CS02:0 = 0), то таймер останавливается. Однако состояние TCNT0 доступно ЦПУ независимо от того работает синхронизация таймера или нет. Запись в регистр таймера через ЦПУ перекрывает любые действия самого счетчика: сброс или счет, т.е. имеет более высокий приоритет.

Работа

Направление счета - всегда приращение, и никакая очистка по внешним событиям не выполняется. Счетчик просто выходит за границы счета когда он переходит свою максимальную 8- битовую величину (MAX = 0xFF) затем перезапускается с низа (0x00). В нормальном действии флаг Переполнения Таймер/Счетчика (TOV0), будет установлен в том же цикле таймера синхронизации как только TCNT0 становится нулем. Флаг TOV0 в этом случае ведется себя подобно девятому биту, за исключением того что только устанавливается, но не очищается. Тем не менее, объединенное прерыванием переполнения таймера, которое автоматически очищает Флаг TOV0, разрешение таймера может быть повышено программным путем. Новое значение таймера может быть записано в любое время.

Временные диаграммы таймера-счетчика 0

Представленные диаграммы соответствуют синхронному по отношению к системной частоте режиму тактирования таймера-счетчика.

На рисунке 28 представлена временная диаграмма работы таймер-счетчика 0 без предделения, при этом, тактовый сигнал таймера (clkT0) показан как сигнал разрешения синхронизации. Счетная последовательность показана в области максимального значения счетчика (0xFF). На рисунках также иллюстрируются моменты установки флагов прерываний.

Рисунок 28. Временная диаграмма таймера-счетчика без предделения

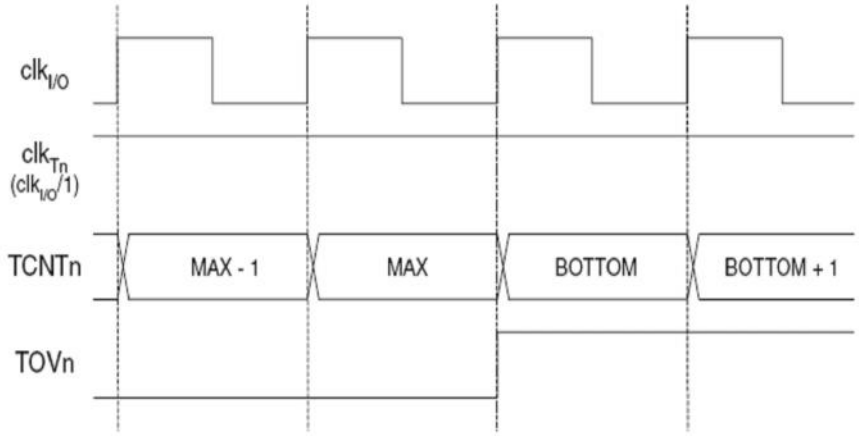
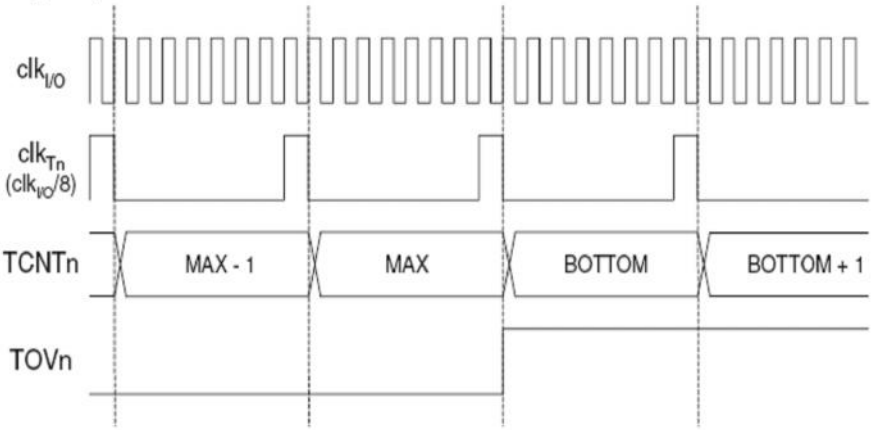


Рисунок 29. Временная диаграмма таймера-счетчика с предделением на 8 (fclk_I/O/8)



8-разрядный Таймер/Счетчик
Описание регистров

Регистр управления таймером-счетчиком 0 - TCCR0

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Бит 2:0 – CS02:0: Выбор синхронизации
С помощью трех настроечных бит имеется возможность выбрать различные тактовые частоты, кратные исходной частоте синхронизации (см. табл. 34).
Таблица 34. Выбор частоты синхронизации таймера 0

CS02	CS01	CS00	Описание
0	0	0	Нет синхронизации. Таймер-счетчик 0 оставлен.
0	0	1	clkT0S/1 (без предделения)
0	1	0	clkT0S/8 (с предделением)
0	1	1	clkT0S/32 (с предделением)
1	0	0	clkT0S/64 (с предделением)
1	0	1	clkT0S/128 (с предделением)
1	1	0	clkT0S/256 (с предделением)
1	1	1	clkT0S/1024 (с предделением)

Если используется внешний режим тактирования для Таймер/Счетчика 0, то переходы на контакте T0 синхронизируют счетчик, даже если контакт будет сконфигурирован как вывод. Эта функция позволяет управлять программным обеспечением подсчета.

Регистр таймера-счетчика - TCNT0

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Регистр Таймера/Счетчика обеспечивает прямой доступ, и для чтения и для операций записи, к 8-разрядному модулю Таймера/Счетчика счетчика.

Регистр Маски Прерывания таймера/Счетчика – TIMSK

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Бит 0 – TOIE0: Включение Прерывания Переполнения Таймер/Счетчика 0
Когда бит TOIE0 устанавливается (1), и I-bit в Регистре Состояния устанавливается (1), включается прерывание переполнения Таймер/Счетчика 0. Соответствующее прерывание выполняется если происходит переполнение в Таймер/Счетчик 0, то есть, когда бит TOV0 устанавливается в регистре флага прерывания таймера/Счетчика – TIFR.

Регистр Флага Прерывания таймера/Счетчика – TIFR

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Бит 0 – TOV0: Флаг Переполнения Таймер/Счетчика 0

Когда происходит переполнение Timer/Counter0 разрядTOV0 устанавливается (1). TOV0 очищается аппаратными средствами, выполняя соответствующий Вектор Обработки прерывания. Альтернативно,TOV0 очищается при записи логической 1 во флаг. Когда SREG I-bit, TOIE0 (Включение прерывания переполнения Таймер/Счетчика0), и TOV0 устанавливаются (один), выполняется прерывание переполнения Таймер/Счетчика 0.

Предделитель таймера-счетчика 0 таймера-счетчика 1

Таймер/Счетчик 1 и Таймер/Счетчик 0 совместно используют тот же самый модуль делителя частоты, но у Таймеров/Счетчиков могут быть различные настройки делителя частоты. Описание ниже применяется к и Таймер/Счетчик1 и Таймер/Счетчик 0.

Внутренний Источник синхронизации

Таймер/Счетчик может быть синхронизирован непосредственно системными часами (устанавливая CSn2:0 = 1). Это обеспечивает самую быструю работу, с максимальной частотой Таймера/Счетчика, равной системной частоте синхронизации (fCLK_I/O). Альтернативно, одно из четырех значений от делителя частоты может использоваться в качестве источника синхронизации. У предварительно масштабируемых делителя частоты есть следующие установки: fCLK_I/O/8, fCLK_I/O/64, fCLK_I/O/256, или fCLK_I/O/1024.

Сброс делителя частоты

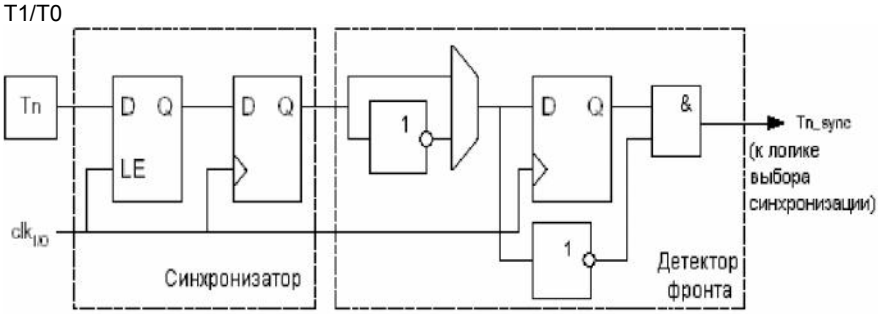
Предделитель является самым простым нереверсивным счетчиком, т.е. работает независимо от логики выбора синхронизации таймера-счетчика и является общим для таймеров 1, 2 и 3. Поскольку логика выбора синхронизации не влияет на таймер-счетчик, то в случае использования предделителя его состояние будет неопределенным. Как пример можно привести неопределенность, которая возникает после разрешения работы таймера, тактируемого через предделитель с настройкой (6 > CSn2:0 > 1). Количество системных тактов с момента разрешения работы таймера до возникновения первого счетного импульса может быть от 1 до N+1, где N – коэффициент деления предделителя (8, 64, 256 или 1024).

Имеется возможность выполнить программный сброс предделителя для синхронизации его работы с таймером. Однако следует учитывать возможность негативного влияния на работу остальных таймеров, которые используют этот же предделитель.

Внешний тактовый источник

Внешний сигнал, подключенный к выводу T1/T0 , может использоваться как тактовый для таймеров- счетчиков (clkT1/clkT0). Вывод T1/T0 опрашивается каждый такт системной синхронизации логикой синхронизации данного вывода. Считанный таким образом сигнал проходит через детектор фронта. На рисунке 30 представлена функциональная схема синхронизации T1/T0 и логики детектора фронта. Регистры тактируются положительным фронтом внутренней системной синхронизации (clkI/O). Детектор фронта генерирует один тактовый импульс clkT1/clkT0 при определении положительного (CSn2:0 = 7) или отрицательного (CSn2:0 = 6) фронта.

Рисунок 30. Функциональная схема синхронизатора и детектора фронта вывода T1/T0

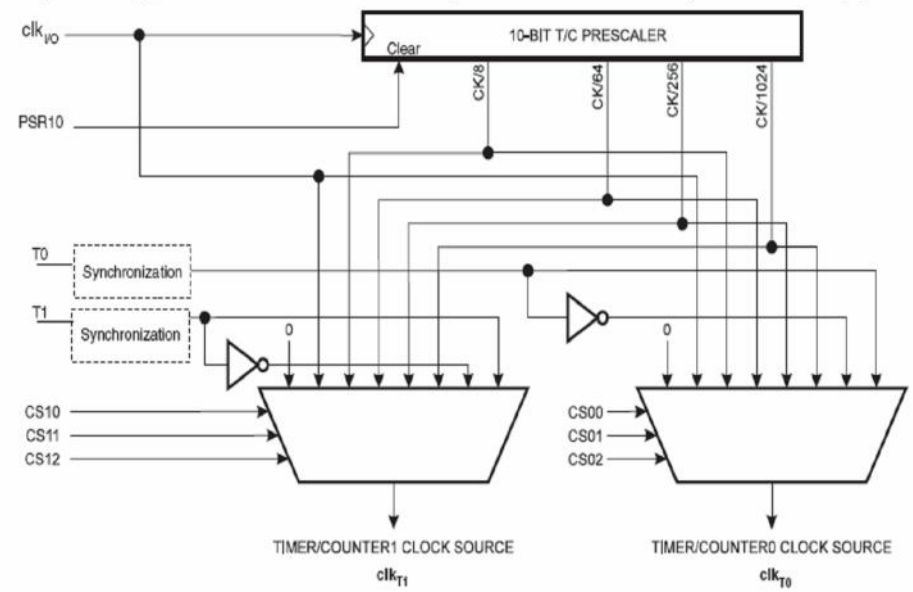


Работа логики синхронизатора и детектора фронта связана с задержкой исходного фронта на выводе T1/T0 на 2.5…3.5 такта системной синхронизации до появления счетного импульса.

Разрешение и запрет тактового входа необходимо выполнять, когда T1/T0 находится в устойчивом состоянии в течение не менее одного такта системной синхронизации, иначе имеется риск генерации ложного тактового импульса синхронизации таймера-счетчика.

Для корректной работы логики преобразования каждый полупериод внешнего тактового сигнала должен быть больше одного периода системной синхронизации. Таким образом, внешний тактовый сигнал должен быть меандром (скважность 2) с частотой минимум вдвое меньшей системной (fExtClk < fclk_I/O/2). Т.к. детектор фронта использует преобразование, то максимальная частота, которую он может определить, равна половине частоты преобразования (теорема преобразования Найквиста). Однако, вследствие изменения частоты системной синхронизации и скважности, вызванных погрешностями тактового генератора (погрешности кварцевого резонатора, керамического резонатора или конденсаторов) рекомендуется, чтобы максимальная частота внешнего тактового сигнала была не более fclk_I/O/2.5.
Частота внешнего тактового сигнала не может быть поделена внутренним предделителем.

Рисунок 31. Делитель частоты для Таймера/Счетчика 0 и Таймера/Счетчика1 (1)



Прим.:1. логика синхронизации на входах T1/T0 показана на рисунке 30.

Регистр специальных функций ввода-вывода – SFIOR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Бит 0 – PSR10: Сброс предделителя таймеров-счетчиков 1,0
Когда этот бит будет равен лог. 1, то предделитель таймеров-счетчиков 1 и 0 будут сброшены. Бит будет очищен аппаратными средствами после того, как работа будет выполнена. Запись нуля к этому биту не будет иметь никакого эффекта. Отметьте, что Таймер/Счетчик 1 и Таймер/Счетчик 0 совместно используют тот же самый делитель частоты, и сброс этого делителя частоты будет влиять на оба таймера. Этот бит будет всегда читаться как нуль.

16-разр. таймер-счетчик 1

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

Основные отличительные особенности:

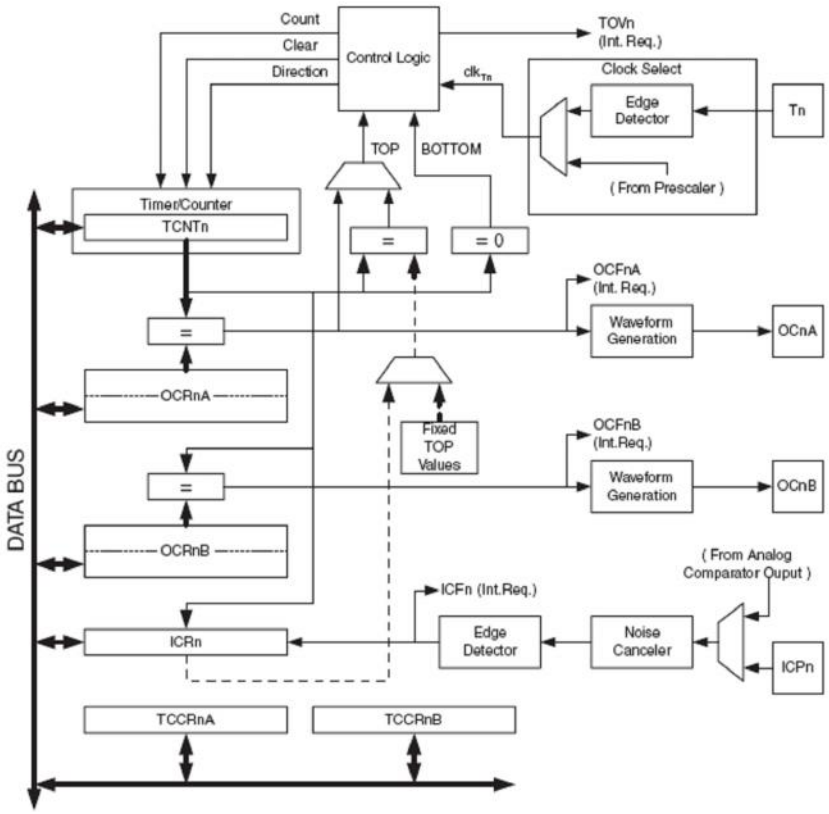
- **16-разрядный счетчик (в т.ч. возможность организации 16-разр. ШИМ)**
- **Два раздельных блока сравнения**
- **Двойная буферизация регистров порога сравнения (OCR)**
- **Один блок захвата**
- **Подавитель шума на входе блока захвата**
- **Режим сброса таймера при совпадении с порогом сравнения (автоматическая перезагрузка)**
- **Широтно-импульсная модуляция без генерации ложных импульсов и фазовая коррекция**
- **Переменный период ШИМ**
- **Частотный генератор**
- **Счетчик внешних событий**
- **Четыре Независимых Источника Прерывания (TOV1, OCF1A, OCF1B, и ICF1)**

Краткий обзор

Большинство наименований регистров и разрядных ссылок в этом разделе пишутся в общей форме. Так индекс “n” заменяет номер таймера-счетчика (1), а “x” заменяет наименование канала сравнения . Однако при программировании необходимо использовать фактические номера и наименования. Например, для записи нового состояния таймера-счетчика 1 в программе необходимо указывать TCNT1.

Укрупненная функциональная схема 16-разр. таймера-счетчика показана на рисунке 32. Если требуется конкретизировать расположение того или иного вывода см. “Расположение выводов”. Регистры ввода-вывода, а также биты или линии ввода-вывода, к которым организован доступ от ЦПУ, выделены жирной линией. Описание регистров, расположение и назначение бит данных таймера представлены в параграфе “Описание регистров 16-разр. таймеров-счетчиков”. на странице 97.

Рисунок 32. 16-разрядная Блок-схема Таймера/Счетчика (1)



Прим.1: Расположение и назначение выводов таймера-счетчика 1 см. на странице 2, Таблице 22 на странице 58, и Таблице 28 на странице 63

Регистры

Регистр таймера-счетчика (TCNT1), регистры порогов сравнения (OCR1A/B), а также регистр захвата (ICR1) являются 16-разрядными регистрами. В связи с этим, во время доступа к этим регистрам должна быть соблюдена специальная процедура (см. **Доступ к 16-разр. регистрам** на странице 79.) Регистры управления таймером (TCCR1A/B) являются 8-разр. регистрами, поэтому, доступ к ним со стороны ЦПУ не связан с какими-либо ограничениями. Все сигналы запросов на прерывание представлены в регистре флагов прерываний таймеров (TIFR) и регистре флагов расширенных прерываний (ETIFR). Все прерывания индивидуально маскируются регистром макси прерываний таймеров (TIMSK) . Регистры TIFR и TIMSK не представлены на функциональной схеме, т.к. они совместно используются другими таймерами микроконтроллера.

Таймер-счетчик может тактироваться внутренне через предделитель или внешне тактовым источником, подключенным к выводу T1. Блок выбора тактового источника позволяет выбрать тактовый источник и фронт, по которому будет изменяться состояние таймера-счетчика. Если тактовый источник не задан, то таймер-счетчик находится в неактивном состоянии. Сигнал на выходе блока выбора тактового источника является тактовым сигналом таймера (clkT1).

Значение регистров порогов сравнения (OCR1A/B) непрерывно сравнивается со значением счетчика. Результат сравнения может использоваться для генерации прямоугольных импульсов с ШИМ или с переменной частотой на выходах OC1A/B. См. также “Блоки сравнения”. странице 85. В случае определения совпадения значений сравниваемых регистров устанавливается соответствующий флаг прерываний (OCF1A/B), который в свою очередь может служить источником прерывания.

Регистр захвата позволяет запомнить состояние таймера-счетчика при возникновении заданного внешнего события (фронт внешнего сигнала) на входе захвата фронта ICP1 или на выводах аналогового компаратора (см. “Аналоговый компаратор” на странице 193). На входе захвата фронта предусмотрена схема цифровой фильтрации (подавитель шума) для снижения риска срабатывания схемы захвата от помехи.
Верхний предел или максимальное значение таймера- счетчика в зависимости от режима работы таймера могут определяться значением в OCR1A, ICR1 или иметь фиксированные значения. Если OCR1A задает верхний предел счета в режиме ШИМ, то он не может использоваться для генерации ШИМ-сигналов. Однако верхний предел в этом случае имеет двойную буферизацию, тем самым допуская изменение его значения в произвольный момент времени. Если верхний предел счета является постоянным значением, то альтернативно можно использовать регистр ICR1, освобождая регистр OCR1A для функции широтно-импульсной модуляции.

Определения

Некоторые определения и их сокращенные наименования, которые интенсивно используются в этом разделе, представлены в таблице 35

Таблица 35. Определения

НП (нижний предел)	Счетчик достигает нулевого значения (0x0000)
МАКС (максимальное значение)	Счетчик достигает максимального значения 0xFFFF (десятич. 65535)
ВП (верхний предел)	Счетчик достигает верхнего предела счета (вершина счета). В качестве вершины счета могут выступать фиксированные значения 0x00FF, 0x01FF, 0x03FF или содержимое регистров OCRnA или ICRn.

Совместимость

По сравнению с предыдущими версиями 16-разр. таймеров-счетчиков данные таймеры доработаны и улучшены. Совместимость этих таймеров соблюдается по следующим позициям:
- Адреса всех регистров, связанных с 16-разр. таймером-счетчиком, в т.ч. регистры прерываний таймеров.
- Расположение бит внутри всех регистров 16-разр. таймеров, в т.ч. регистры прерываний таймеров
- Векторы прерываний

У следующих управляющих бит изменены наименования, но сохранено назначение и расположение в регистре:
- PWM10 заменен на WGM10.
- PWM11 заменен на WGM11.
- CTC1 заменен на WGM12.

Следующие биты добавляются к 16-разрядному Регистру команд Таймера/Счетчика:
- FOC1A и FOC1B добавляются к TCCR1A.
- WGM13 добавляется к TCCR1B.

У 16-разрядного Таймера/Счетчика есть улучшения, которые будут влиять на совместимость в некоторых особых случаях.

Доступ к 16-разрядным Регистрам

Регистры TCNT1, OCR1A/B и ICR1 являются 16-разрядными, поэтому, доступ к ним через 8- разр. шину данных AVR ЦПУ может быть осуществлен с помощью двух инструкций чтения или записи. У каждого 16-разр. таймера имеется свой 8-разр. регистр для временного хранения старшего байта данных. Поэтому, во время доступа к 16-разр. регистрам одного таймера используется один и тот же временный регистр. Чтение/запись младшего байта инициирует 16- разр. операцию чтения/записи. Если выполняется запись младшего байта 16-разр. регистра, то за один такт ЦПУ одновременно записываются и младший байт и старший байт из временного регистра. Если выполняется чтение младшего байта 16-разр. регистра, то за один такт ЦПУ параллельно с чтением младшего байта происходит копирование старшего байта 16-регистра во временный регистр.

Не все 16-разрядные регистры используют временный регистр для копирования старшего байта. Чтение 16-разр. регистров OCR1A/B не связано с использованием временного регистра. Таким образом, чтобы записать данные в 16-разр. регистр, необходимо сначала записать старший байт, а затем младший. А при чтении 16-разр. регистра, наоборот, сначала считывается младший байт, а затем старший. Ниже приведен пример на Ассемблере и Си, показывающие как осуществлять доступ к 16-разр. регистрам таймера. В примере предполагается, что во время обновления временного регистра не возникает прерываний. Аналогично может быть выполнен доступ к регистрам OCR1A/B и ICR1.
Отметьте, что при использовании “C”, компилятор обрабатывает 16-разрядный доступ.

Пример кода на Ассемблере(1)

```
...
; Установка TCNTn = 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNTnH,r17
out TCNTnL,r16
; Чтение TCNTn в r17:r16
in r16,TCNTnL
in r17,TCNTnH
...
```

Пример кода на Си(1)

```
unsigned int i;
...
/* Установка TCNTn = 0x01FF */
TCNTn = 0x1FF;
/* Чтение TCNTn в i */
i = TCNTn;
...
```

Примечание: 1. См. “О Примерах кода” на странице 8.

В примере на Ассемблере значение TCNT1 возвращается парой регистров r17:r16. При этом следует обратить внимание на проблему, которая связана с необходимостью выполнения двух инструкций для получения доступа к 16-разр. регистру. Если после выполнения первой инструкции доступа 16-разр. регистра происходит прерывание и в процедуре обработки прерывания также происходит обновление этого же или другого регистра, но относящегося к тому же таймеру, то по завершении обработки прерывания изменяется содержимое временного регистра и выполнение второй инструкции приведет к некорректному результату. Таким образом, когда и в основной программе и в прерываниях происходит обновление временного регистра, то в основной программе перед инициацией доступа к 16-разр. регистру необходимо запретить прерывания.

В следующем примере показано как корректно выполнить чтение регистра TCNT1 без опасности изменения содержимого временного регистра в прерываниях. Аналогично данный пример следует распространять на доступ к регистрам OCR1A/B и ICR1.

Пример кода на Ассемблере (1)

```
TIM16_ReadTCNTn:
; Запомнили состояние общего флага прерываний
in r18,SREG
; Запрет прерываний
cli
; Чтение TCNTn в r17:r16
in r16,TCNTnL
in r17,TCNTnH
; Восстановили состояние общего флага прерываний out SREG,r18
ret
```

Пример кода на Си (1)

```
unsigned int TIM16_ReadTCNTn( void )
{
unsigned char sreg;
unsigned int i;
/* Запомнили состояние общего флага прерываний */
sreg = SREG;
/* Запретили прерывания */
_CLI();
/* Чтение TCNTn в i */
i = TCNTn;
/* Восстановили состояние общего флага прерываний */
SREG = sreg;
return i;
}
```

Примечание: 1. См. “О Примерах кода” на странице 8

В коде на Ассемблере значение регистра TCNT1 возвращается парой регистров r17:r16.

В следующем примере показано как избежать опасного влияния изменения содержимого временного регистра при возникновении прерывания во время записи в регистр TCNT1. На этом же принципе может быть выполнена запись в регистры OCR1A/B/C или ICR1

Пример кода на Ассемблере (1)

```
TIM16_WriteTCNTn:
; Запомнили состояние общего флага прерываний
in r18,SREG
; Запрет прерываний
cli
; Копирование TCNTn в r17:r16
out TCNTnH,r17
out TCNTnL,r16
89
; Восстановили состояние общего флага прерываний
out SREG,r18
ret
```

Пример кода на Си (1)

```
void TIM16_WriteTCNTn( unsigned int i )
{
unsigned char sreg;
unsigned int i;
/* Запомнили состояние общего флага прерываний */
sreg = SREG;
/* Запрет прерываний */
_CLI();
/* Копирование TCNTn в i */
TCNTn = i;
/* Восстановили состояние общего флага прерываний */
SREG = sreg;
}
```

Примечание: 1. См. “О Примерах кода” на странице 8

В примере на Ассемблере предполагается, что записываемое значение в TCNT1 предварительно записано в пару регистров r17:r16.

Повторное использование временного регистра старшего байта

Если выполняется запись в несколько 16-разр. регистров и при этом значение старшего байта одинаково для всех регистров, то достаточно однократно выполнить запись старшего байта. Однако при этом также учтите возможность некорректного завершения такой операции, если используются прерывания (см. выше описание механизма разрешения такой проблемы).

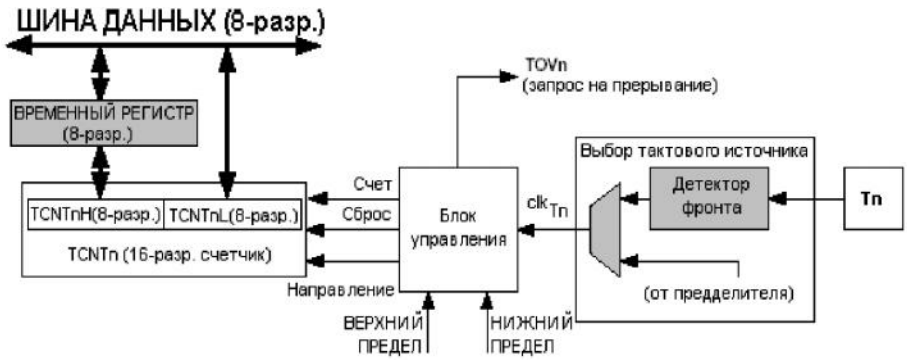
Тактовые источники таймера-счетчика 1

Таймер-счетчик может использовать как внешний, так и внутренний тактовые сигналы. Источник тактового сигнала выбирается соответствующей схемой микроконтроллера под управлением бит выбора синхронизации (CS12:0), которые находятся в регистре B управления таймером-счетчиком (TCCR1B). Более подробная информация по тактовым источникам и предделителю приведена в разделе "Предделители таймера-счетчика 0 и таймера-счетчика 1" на странице 74.

Блок счетчика

Основным элементом 16-разр. таймера-счетчика является программируемый реверсивный 16- разрядный счетчик. На рисунке 33 представлена функциональная схема счетчика и окружающих его элементов.

Рисунок 33 Функциональная схема счетчика



Описание внутренних сигналов:

Счет – Инкрементирует или декрементирует состояние TCNT1 на 1.
Направление – Задает прямой счет (инкрементирование) или обратный счет (декрементирование).
Сброс – Сброс TCNT1 (установка всех разрядов к лог. 0).
clkTn – Синхронизация таймера-счетчика 1.
Верхний предел – Сигнализирует о достижении TCNT1 максимального значения.
Нижний предел – Сигнализирует о достижении TCNT1 минимального значения (нуля).

Содержимое 16-разр. счетчика разбито на две 8-разр. ячейки, расположенных в памяти ввода- вывода: Старший байт счетчика (TCNT1H) , в котором хранятся старшие 8-разрядов счетчика, и младший байт счетчика (TCNT1L), в котором хранятся младшие 8-разрядов. ЦПУ не имеет непосредственного доступа к регистру TCNT1H. Если ЦПУ выполняет доступ к TCNT1H, то фактически обращение происходит к временному регистру. Во временный регистр копируется значение TCNT1H, если выполняется чтение регистра TCNT1L и в TCNT1H копируется содержимое временного регистра, если выполняется запись в TCNT1L. Такой механизм реализован для считывания/записи 16-разр. значения счетчика за один такт ЦПУ в условиях 8-разр. шины данных. Следует обратить внимание, что в некоторых случаях запись в регистр TCNT1 во время счета счетчиком будет давать непредсказуемый результат. Такие случаи описаны в последующих параграфах.

В зависимости от используемого режима работы каждый такт синхронизации таймера clkT1 счетчик будет сбрасываться, инкрементироваться или декрементироваться. Сигнал clkT1 может быть внешним или внутренним, что задается битами выбора синхронизации (CS12:0). Если тактовый источник не задан (CS12:0 = 0), то таймер останавливается. Однако содержимое TCNT1 остается доступным ЦПУ независимо от наличия синхронизации на clkT1. Если ЦПУ выполняет запись в TCNT1, то тем самым блокируется (запись имеет более высокий приоритет) любое действие счетчика: сброс или счет.

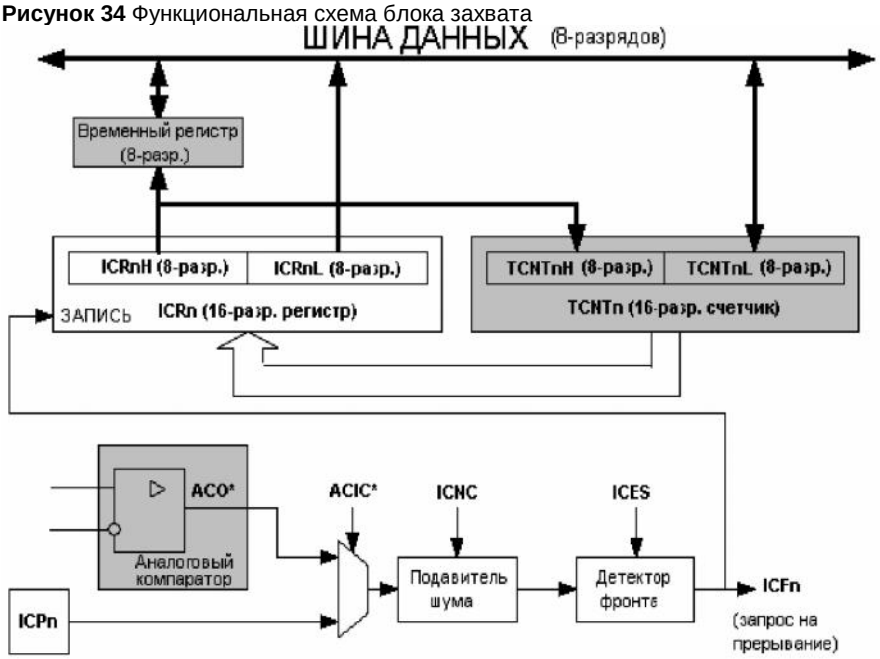
Алгоритм счета определяется значением бит режима работы таймера (WGM13:0), расположенных в регистрах A и B управления таймером-счетчиком (TCCR1A и TCCR1B). Имеется четкая связь между алгоритмом счета счетчика и формой генерируемого на выходе OC1x сигнала. Боле подробная информация об этом приведена в “Режимы работы 16-разр. таймеров-счетчиков” на странице 88. Установка флага переполнения таймера-счетчика (TOV1) происходит в зависимости выбранного с помощью бит WGM13:0 режима работы. Флаг TOV1 может использоваться для генерации прерывания ЦПУ.

Блок захвата

Таймер-счетчик содержит блок захвата, который запоминает состояние счетчика при возникновении внешнего события, тем самым определяя время его возникновения. В качестве события/событий выступает внешний сигнал, подключенный к выводу ICP1. Для таймера-счетчика 1 альтернативно может использоваться аналоговый компаратор в качестве источника внешнего события. Результат захвата состояния таймера может использоваться для вычисления частоты, скважности импульсов и других параметров импульсных сигналов. Альтернативно это значение может использоваться для создания журнала событий.

Функциональная схема блока захвата представлена на рисунке 34. Некоторые блоки на функциональной схеме, которые не относятся напрямую к блоку захвата, залиты серым цветом. Символ “n” в наименованиях бит и регистров заменяет номер таймера-счетчика.

Рисунок 34 Функциональная схема блока захвата



Если на входе захвата (ICP1) или альтернативно на выходе аналогового компаратора (ACO) возникает изменение логического уровня (событие), которое соответствует установкам детектора фронта, то выполняется захват состояния таймера. При этом 16-разр. значение содержимого таймера (TCNT1) помещается в регистр захвата (ICR1). Флаг захвата (ICF1) устанавливается на том же такте ЦПУ, на котором произошло копирование значения TCNT1 в ICR1. Установка бита TICIE1 =1 разрешает прерывание по установке флага захвата. Флаг ICF1 автоматически сбрасывается при переходе на вектор прерывания. Альтернативно флаг ICF1 сбрасывается программно, если записать в него лог. 1.

Считывание 16-разр. значения регистра захвата (ICR1)
выполняется чтением сначала младшего байта (ICR1L), а затем старшего байта (ICR1H). При выполнении команды чтения младшего байта значение старшего байта автоматически копируется во временный регистр. Если ЦПУ выполняет команду чтения регистра ICR1H, то фактически считывается содержимое временного регистра. Запись в регистр ICR1 возможна только в том случае, если битами задания режима работы таймера выбран режим, в котором значение регистра ICR1 задает верхний предел счета. В этом случае необходимо выполнить соответствующую установку бит режима работы (WGM13:0), а только затем выполнить запись значения верхнего предела в регистр ICR1. Запись 16-разр. значения в регистр ICR1 выполняется путем записи сначала старшего байта в ICR1H, а только затем младшего байта в ICR1L (см. также “Доступ к 16- разр. регистрам”).

Источник срабатывания механизма захвата

Основным источником, инициирующим захват состояния таймер-счетчика, является вывод захвата (ICP1). Таймер-счетчик 1 также альтернативно может использовать выход аналогового компаратора в качестве источника инициации захвата. Для этого необходимо установить бит разрешения захвата аналоговым компаратором (ACIC) в регистре состояния и управления аналогового компаратора (ACSR). Учтите, что изменение источника инициации захвата может привести к возникновению захвата. Поэтому, после изменения источника должен быть сброшен флаг захвата.

Входные каскады, связанные с выводом захвата (ICP1) и выходом аналогового компаратора (ACO) выполнены по аналогии с каскадом, связанного с выв. T1 (см. рис. 30). Детекторы фронта также идентичны. Однако если разрешена работа подавителя шумов, то последовательно перед детектором фронта включается дополнительная логика, которая реагирует на изменение уровня, если он находился на постоянном уровне не менее 4 тактов ЦПУ. Обратите внимание, что вход подавителя шумов и детектора фронта всегда разрешен, если таймер-счетчик находится в режиме, где ICR1 определяет вершину счета.

Захват можно инициировать программно путем управления настройками порта на выводе ICP1.

Подавитель шума

Подавитель шума позволяет повысить помехозащищенность за счет использования простой схемы цифровой фильтрации. Для изменения выходного состояния подавителя шума необходимо, чтобы на его входе совпало значение четырех выборок.
Работа подавителя шума разрешается путем установки бита разрешения подавления шумов на входе захвата (ICNC1) в регистре B управления таймером-счетчиком (TCCR1B). С разрешением работы подавителя шума вводится задержка на 4 такта между возникновением изменения входного уровня и собственно захватом состояния таймера. Подавитель шума использует системную синхронизацию и, следовательно, не зависит от настройки предделителя таймера.

Использование блока захвата

Основная проблема при использовании блока захвата состоит в необходимости выделения достаточного процессорного времени на обработку возникшего события. Время между двумя событиями является критичным параметром. Если процессор не успевает считать содержимое ICR1 до момента возникновения следующего события, то регистр ICR1 обновит свое содержимое. В этом случае результат захвата будет некорректным.

Если используется прерывание по захвату, то в процедуре обработки прерываний регистр ICR1 необходимо считать как можно раньше. Даже при том, что прерывание по захвату имеет относительно высокий приоритет, время реакции на прерывание будет зависеть от максимального числа тактов, которое требуется на выполнение процедуры обработки любого другого прерывания.

Использование блока захвата не рекомендуется, если таймер используется в режиме, когда значение верхнего предела счета (разрешающая способность) активно обновляется в процессе работы.
Для измерения скважности (или коэффициента заполнения) импульсов необходимо после каждого захвата изменять фронт, по которому происходит захват. Данное изменение необходимо выполнять как можно раньше после считывания регистра ICR1. После изменения фронта необходимо сбросить флаг захвата ICF1 записью в него лог. 1.

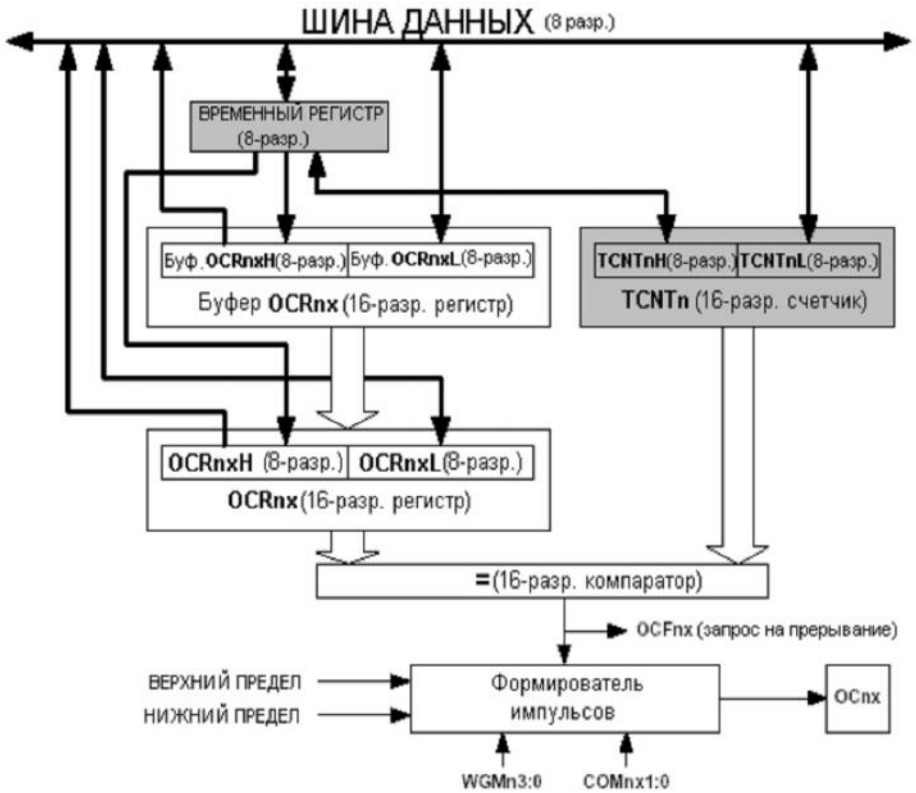
Если требуется измерение только частоты, то сброс флага ICF1 не требуется (если используется обработка по прерыванию).

Блоки сравнения

16-разрядный цифровой компаратор непрерывно сравнивает значение TCNT1 со значением регистра порога сравнения (OCR1x). Если значение TCNT равно OCR1x, то компаратор формирует сигнал совпадения (равенства). Следующий за совпадением такт ЦПУ устанавливает флаг сравнения (OCF1x). Если бит OCIE1x = 1, то установка флага сравнения приведет к генерации прерывания по результату сравнения. Флаг OCF1x автоматически сбрасывается после перехода на вектор обработки прерывания. Альтернативно флаг OCF1x сбрасывается программно, если записать в него лог. 1. Сигнал совпадения используется формирователем выходного сигнала, результирующая форма которого зависит от выбранного с помощью бит WGM13:0 режима работы таймера и режима формирования импульсов (биты COM1x1:0). Сигналы ВЕРХНИЙ ПРЕДЕЛ и НИЖНИЙ ПРЕДЕЛ используются формирователем импульсов для отработки особых случаев задания экстремальных значений в некоторых режимах работы (см. “Режимы работы 16-разр. таймеров-счетчиков” на странице 88).

У канала сравнения A имеется своя отличительная особенность, которая позволяет задать верхний предел счета (т.е. разрешающую способность счетчика). В дополнение к разрешающей способности верхний предел определяет период формируемых импульсов.
На рисунке 35 показана функциональная схема блока сравнения. Символ “n” в наименованиях бит и регистров заменяет номер таймера (n = 1 для Таймера/Счетчика 1), а “x” заменяет наименование канала сравнения (A/B).
Те блоки на функциональной схеме, которые не относятся к блоку сравнения, залиты серым цветом.

Рисунок 35. Блок-схема Модуля сравнения



Если задан любой из 12 режимов широтно-импульсной модуляции (ШИМ), то в этом случае регистр OCR1x содержит двойную буферизацию. Если таймер работает в нормальном режиме или режиме сброса при совпадении (CTC), то двойная буферизация отключается.

Двойная буферизация синхронизирует обновление регистра порога сравнения OCR1x по достижении верхнего или нижнего предела счета в зависимости от выбранного режима работы (алгоритма счета). Такая синхронизация предотвращает возможность возникновения несимметричных ШИМ- импульсов нечетной длины, тем самым гарантируя отсутствие сбоев при генерации прямоугольных импульсов.

Доступ к регистру OCR1x хоть и кажется сложным, но выполнен таким образом не случайно. Если двойная буферизация разрешена, то ЦПУ фактически осуществляет доступ к буферному регистру OCR1x. Если же двойная буферизация отключена, то ЦПУ обращается к регистру OCR1x непосредственно. Содержимое регистра OCR1x (в т.ч. и буферного) может измениться только путем непосредственной записи в него (таймер-счетчик не обновляет содержимое данного регистра автоматически аналогично регистрам TCNT1 и ICR1). Таким образом, OCR1x считывается напрямую, а не через временный регистр старшего байта. Запись регистров OCR1x происходит через временный регистр, т.к. все 16 разр. участвуют в сравнении непрерывно. Первым необходимо записать старший байт OCR1xH. Если выполняется запись по адресу старшего байта, то содержимое временного регистра обновляется записываемым значением. Если выполняется запись младшего байта (OCR1xL), то параллельно копируется содержимое временного регистра в старшие 8-разрядов буферного регистра OCR1x или регистра порога сравнения OCR1x, тем самым обновляя все 16 разрядов за один такт ЦПУ (см. также "Доступ к 16- разр. регистрам" на странице 79).

Принудительная установка результата сравнения

В режимах генерации импульсов без ШИМ в формирователе импульсов результат сравнения может быть установлен непосредственно через бит принудительной установки результата сравнения FOC1x. Принудительная установка результата сравнения не приводит к установке флага OCF1x или сбросу/перезагрузке таймера, но влияет на состояние вывода OC1x, который будет устанавливаться, сбрасываться или переключаться (инвертироваться) в зависимости от выбранной установки бит COM1x1:0

Результат сравнения блокируется записью в TCNT1

Если ЦПУ осуществляет запись в регистр TCNT1, то результат сравнения будет игнорироваться на следующем такте синхронизации таймера, даже если таймер остановлен. Данная функция позволяет установить в регистре OCR1x то же значение, что и в TCNT1 без генерации запроса на прерывание, если разрешено тактирование таймера-счетчика.

Использование блока сравнения

Поскольку запись в TCNT1 в любом из режимов работы блокирует отработку совпадения на один такт синхронизации таймера, то имеются некоторые опасные ситуации при изменении TCNT1 во время использования любого из каналов сравнения, независимо работает таймер-счетчик или нет. Если в TCNT1 записано значение равное OCR1x, то возникающее совпадение игнорируется, тем самым вызывая некорректную генерацию импульсов. Следует избегать записи в TCNT1 значения равного верхнему пределу в ШИМ-режимах с переменным значением верхнего предела. В этом случае совпадение по достижении верхнего предела игнорируется и счет продолжится до 0xFFFF. Аналогично, следует избегать записи в TCNT1 значения равного нижнему пределу, если счетчик работает как вычитающий (обратный счет).

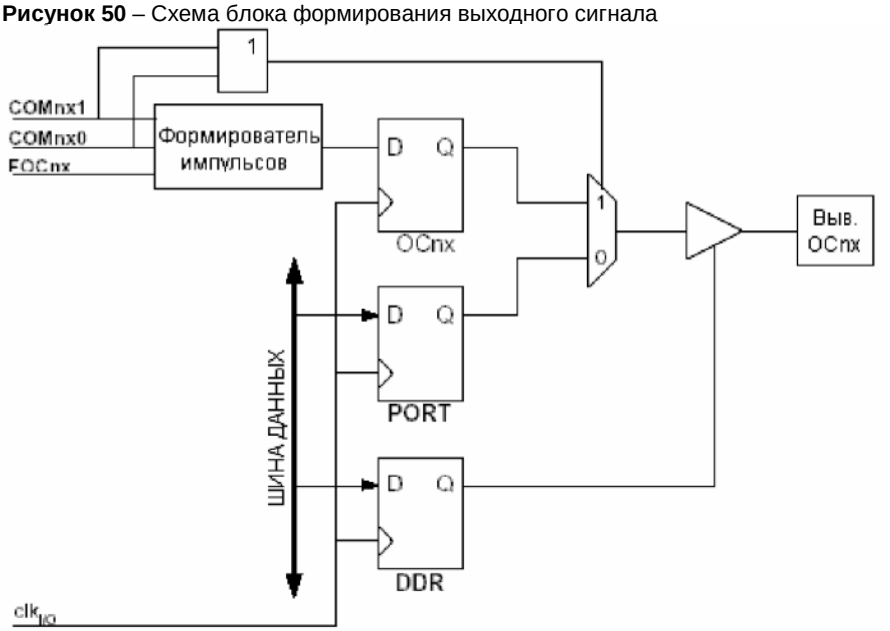
Прежде чем настроить вывод OC1x на вывод в регистре направления данных необходимо выполнить инициализацию регистра OC1x. Самым простым способом решения этой задачи является использование бита принудительной установки результата сравнения (FOC1x) при работе таймера в нормальном режиме. Регистр OC1x сохраняет свое состояние даже при изменении режима работы таймера.

Учтите, что биты COM1x1:0 не содержат схемы двойной буферизации и любые их изменения вступают в силу мгновенно.

Блок формирования выходного сигнала

Биты задания режима формирования выходного сигнала (COM1x1:0) имеют двойное назначение. С одной стороны биты COM1x1:0 используются формирователем сигнала и определяют какое логическое состояние должно быть на выходе OC1x при возникновении следующего совпадения. С другой стороны, биты COM1x1:0 используются для разрешения/запрета альтернативной функции вывода порта OC1x. На рисунке 36 представлена упрощенная логическая схема, на которую воздействуют биты COM1x1:0. На рисунке показаны только те регистры управления портом ввода-вывода (DDR и PORT), на которые оказывает действие биты COM1x1:0. Если происходит системный сброс, то выход регистра OC1x принимает нулевое состояние.

Рисунок 50 – Схема блока формирования выходного сигнала



Функция линии универсального порта ввода-вывода заменяется на функцию выхода формирователя сигнала OC1x, если хотя бы один из бит COM1x1:0 установлен (логика ИЛИ). Однако, управление направлением вывода OC1x (вход или выход) в этом случае остается за соответствующим битом регистра направления данных (DDR). Чтобы значение регистра OC1x присутствовало на выводе OC1x необходимо настроить данную линию на вывод (установить соотв. бит в DDRB). Управление вводом альтернативной функции не зависит от режима работы таймера за некоторыми исключениями (см. табл. 37,38 и 39).

Схемотехника выходной логики позволяет инициализировать состояние регистра OC1x перед разрешением настройки вывода OC1x в качестве выхода. Обратите внимание, что в некоторых режимах работы имеются зарезервированные состояния бит COM1x1:0. См. “Описание регистров 16-разр. таймеров-счетчиков” на странице 97.Установки бит COM1x1:0 не оказывают никакого влияния на работу блока захвата.

Режимы генерации импульсов

Установки бит COM1x1:0 оказывают различное влияние в зависимости от выбранного режима работы: нормального, сброса при совпадении и ШИМ. Общим для всех режимов работы является не выполнение каких-либо действий с регистром OC1x при возникновении совпадения, если COM1x1:0 = 0. В таблице 36 на странице 97 описано действие различных установок этих бит для режимов без ШИМ. Аналогичная информация для режима с быстрой ШИМ приведена в таблице 37 на странице 98, а для ШИМ с фазовой и частотной коррекцией в таблице 38 на странице 98.
Изменение состояния бит COM1x1:0 вступает в силу при следующем после их записи совпадении. В режимах без ШИМ воздействовать на генерацию импульсов можно с помощью стробирующего бита принудительной установки результата сравнения FOC1x.

Режимы работы

Под режимом работы 16-разр. таймера понимается его алгоритм счета и поведение связанного с ним выхода формирователя импульсов, что определяется комбинацией бит, задающих режим работы таймера (WGM13:0) и режим формирования выходного сигнала (COM1x1:0). При этом биты задания режима формирования выходного сигнала не влияют на алгоритм счета, т.к. алгоритм счета зависит только от состояния бит задания режима работы таймера. В режимах с ШИМ биты COM1x1:0 позволяют включить/отключить инверсию на генерируемом ШИМ-выходе (т.е. выбрать ШИМ с инверсией или ШИМ без инверсии). Для режимов без ШИМ биты COM1x1:0 определяют, какое действие необходимо выполнить при возникновении совпадения: сбросить, установить или инвертировать выход (см. также “Блок формирования выходного сигнала на странице 87” и "Временные диаграммы 16-разр. таймеров-счетчиков" на странице 95).

Нормальный режим работы

Самым простым режимом работы является нормальный режим (WGM13:0 = 0). В данном режиме счетчик работает как суммирующий (инкрементирующий), при этом сброс счетчика не выполняется. Переполнение счетчика происходит при переходе через максимальное 16-разр. значение (0xFFFF) к нижнему пределу счета (0x0000). В нормальном режиме работы флаг переполнения таймера-счетчика TOV1 будет установлен на том же такте синхронизации, когда TCNT1 примет нулевое значение.

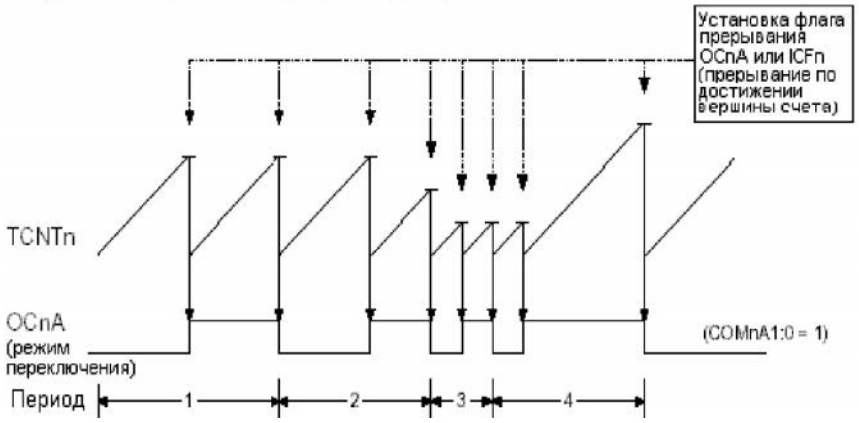
Фактически, флаг переполнения TOV1 является 17-ым битом таймера-счетчика за тем исключением, что он только устанавливается и не сбрасывается. Однако программно это свойство может быть использовано для повышения разрешающей способности таймера, если использовать прерывание по переполнению таймера, при возникновении которого флаг TOV1 сбрасывается автоматически. Для нормального режима работы не существует каких-либо особых ситуаций, поэтому запись нового состояния счетчика может быть выполнена в любой момент. В нормальном режиме можно использовать блок захвата. Однако при этом следует соблюдать, чтобы максимальный интервал времени между возникновениями внешних событий не превысил периода переполнения счетчика. Если такое условие не соблюдается, необходимо использовать прерывание по переполнению таймера-счетчика или предделитель.

Блок сравнения может использоваться для генерации прерываний. Не рекомендуется использовать выход OC1x для генерации сигналов в нормальном режиме работы, т.к. в этом случае будет затрачена значительная часть процессорного времени.

Режим сброса таймера при совпадении (CTC)

В режиме сброса таймера при совпадении (WGM13:0 = 4 или 12) разрешающая способность таймера задается регистрами OCR1A или ICR1. В режиме CTC происходит сброс счетчика (TCNT1), если его значение совпадает со значением регистра OCR1A (WGM13:0 = 4) или с ICR1 (WGM13:0 = 12). Значение регистра OCR1A или ICR1 определяет верхний предел счета, а, следовательно, и разрешающую способность таймера. В данном режиме обеспечивается более широкий диапазон регулировки частоты генерируемых прямоугольных импульсов. Он также упрощает работу счетчика внешних событий. Временная диаграмма работы таймера в режиме CTC показана на рисунке 37. Счетчик (TCNT1) инкрементирует свое состояние до тех пор, пока не возникнет совпадение со значением OCR1A или ICR1, а затем счетчик (TCNT1) сбрасывается.

Рисунок 37 – Временная диаграмма для режима CTC



По достижении верхнего предела счета может генерироваться прерывание с помощью флагов OCF1A или ICF1, соответствующим используемым регистрам для задания верхнего предела счета. Если прерывание разрешено, то процедура обработки прерывания может использоваться для обновления верхнего предела счета. Однако, задание значения вершины счета близкого к значению нижнего предела счета, когда счетчик работает без предделения или с малым значением предделения, необходимо выполнять с особой осторожностью, т.к. в режиме CTC нет двойной буферизации. Если значение, записанное в OCR1A или ICR1, меньше текущего значения TCNT1, то сброс счетчика по условию совпадения наступит, когда он достигнет максимального значения (0xFFFF), затем перейдет в исходное состояние 0x0000 и достигнет нового значения OCR1A или ICR1. Во многих случаях возникновение такой ситуации не желательно. В качестве альтернативы может выступить режим быстрой ШИМ, где регистр OCR1A определяет верхний предел счета (WGM13:0 = 15), т.к. в этом случае OCR1A имеет двойную буферизацию.

Для генерации сигнала в режиме CTC выход OC1A может использоваться для изменения логического уровня при каждом совпадении, для чего необходимо задать режим переключения (COM1A1:0 = 1). Значение OC1A будет присутствовать на выводе порта, только если для данного вывода задано выходное направление(DDR_OC1A = 1). Максимальная частота генерируемого сигнала равна fOC1A = fclk_I/O/2, если OCR1A = 0x0000. Для других значений OCR1 частоту генерируемого сигнала можно определить по формуле:

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

где переменная N задает коэффициент деления предделителя (1, 8, 32, 64, 128, 256 или 1024).

Также как и для нормального режима работы, флаг TOV1 устанавливается на том же такте таймера, когда его значение изменяется с 0xFFFF на 0x0000.

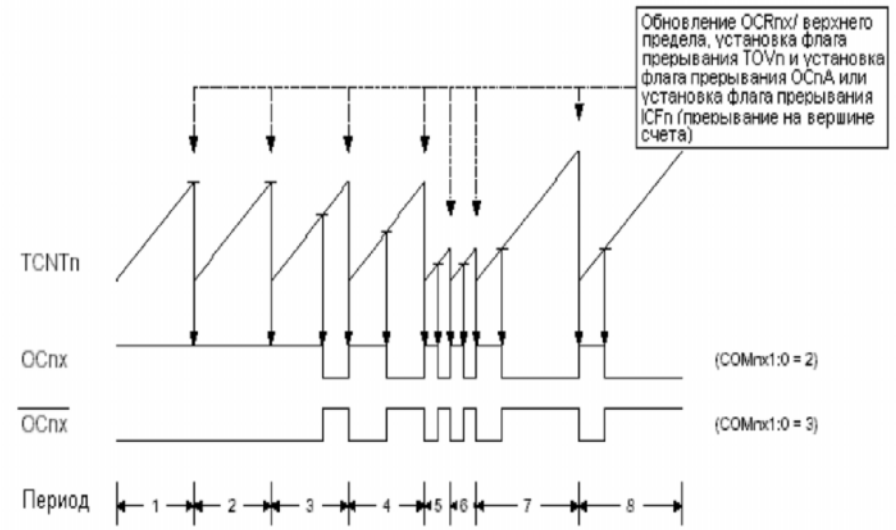
Режим быстрой ШИМ

Режим быстрой широтно-импульсной модуляции (ШИМ) (WGM13:0 = 5, 6, 7, 14, или 15) предназначен для генерации ШИМ-импульсов повышенной частоты. В отличие от других режимов работы в этом используется однонаправленная работа счетчика. Счет выполняется в направлении от нижнего к верхнему пределу счета.
Если задан неинвертирующий режим выхода, то при совпадении TCNT1 и OCR1x сигнал OC1x устанавливается, а на верхнем пределе счета сбрасывается. Если задан инвертирующий режим, то выход OC1x сбрасывается при совпадении и устанавливается на верхнем пределе счета. За счет однонаправленности счета, рабочая частота для данного режима в два раза выше по сравнению с режимом ШИМ с фазовой коррекцией, где используется двунаправленный счет.
Возможность генерации высокочастотных ШИМ сигналов делает использование данного режима полезным в задачах стабилизации питания, выпрямления и цифро-аналогового преобразования. Высокая частота, при этом, позволяет использовать внешние элементы физически малых размеров (индуктивности, конденсаторы), тем самым снижая общую стоимость системы.

Разрешающая способность ШИМ может быть фиксированной 8, 9 или 10 разрядов или задаваться регистром ICR1 или OCR1A, но не менее 2 разрядов (ICR1 или OCR1A = 0x0003) и не более 16 разрядов (ICR1 или OCR1A = 0xFFFF).
Разрешающая способность ШИМ при заданном значении верхнего предела (ВП) вычисляется следующим образом:

$$R_{FPWM} = \frac{\log(TOP+1)}{\log(2)}$$

В режиме быстрой ШИМ счетчик инкрементируется до совпадения его значения с одним из фиксированных значений 0x00FF, 0x01FF или 0x03FF (если WGM13:0 = 5, 6, или 7), значением в ICR1 (если WGM13:0 = 14) или значением в OCR1A (если WGM13:0 = 15), а затем сбрасывается следующим тактом синхронизации таймера. Временная диаграмма для режима быстрой ШИМ представлена на рисунке 38. На рисунке показан режим быстрой ШИМ, когда для задания верхнего предела используется регистр OCR1A или ICR1.
Значение TCNT1 на временной диаграмме показано в виде графика функции для иллюстрации однонаправленности счета. На диаграмме показаны как инвертированный, так и неинвертированный ШИМ-выходы. Короткой горизонтальной линией показаны точки на графике TCNT1, где совпадают значения OCR1x и TCNT1x. Флаг прерывания OC1x устанавливается при возникновении совпадении.
Рисунок 38. Временная диаграмма для режима быстрой ШИМ



Флаг переполнения таймера-счетчика (TOV1) устанавливается всякий раз, когда счетчик достигает верхнего предела. Дополнительно тем же тактовым импульсом вместе с флагом TOV1 могут установиться флаги OC1A или ICF1, если для задания верхнего предела используется регистр OCR1A или ICR1, соответственно. Если одно из этих прерываний разрешено, то в процедуре обработки прерывания может быть выполнено обновление верхнего предела счета и порогов сравнения.

Если изменяется значение верхнего предела счета, то необходимо соблюдение условия, чтобы записываемое новое значение верхнего предела было больше или равно значений во всех регистрах порога сравнения. В противном случае совпадение между TCNT1 и OCR1x никогда не возникнет. Обратите внимание, что при использовании фиксированных значений верхнего предела во время записи в регистры OCR1x происходит маскирование к 0 неиспользуемых разрядов.

Механизм модификации регистра ICR1 отличается от OCR1A в том случае, если он используется для задания верхнего предела. Регистр ICR1 не имеет двойной буферизации. Это означает, что если в ICR1 записывается малое значение во время работы счетчика с малым предделением или без него, то имеется опасность записи в регистр ICR1 значения, которое окажется меньше текущего значения TCNT1. Как результат, в такой ситуации будет пропущено совпадение на вершине счета. В этом случае счетчик дойдет до максимального значения (0xFFFF), перезапустится со значения 0x0000, а только затем возникнет совпадение. Регистр OCR1A содержит схему двойной буферизации, поэтому, его можно модифицировать в любой момент времени.

Если выполняется запись по адресу OCR1A, то фактически значение помещается в буферный регистр OCR1A. Если же возникает совпадение между TCNT1 и вершиной счета, то следующим тактом синхронизации таймера происходит копирование буферного регистра в регистр порога сравнения OCR1A. Обновление регистра выполняется тем же тактом, что и сброс TCNT1 и установка флага TOV1.

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

В режиме быстрой ШИМ блоки сравнения позволяют генерировать ШИМ-сигналы на выводах OC1x. Если COM1x1:0 =2, то задается ШИМ без инверсии выхода, а если COMnx1:0 = 3, то задается режим ШИМ с инверсией на выходе (см. таблицу 37). Фактическое значение OC1x можно наблюдать на выводе порта, если для него задано выходное направление (DDR_OC1x). ШИМ- сигнал генерируется путем установки (сброса) регистра OC1x при возникновении совпадения между OCR1x и TCNT1, а также путем сброса (установки) регистра OC1x вместе со сбросом счетчика (переход с верхнего предела на нижний предел).

Частота ШИМ выходного сигнала для заданного значения верхнего предела (ВП) определяется выражением:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

где N – переменная, которая задает значение коэффициента предделения (1, 8, 32, 64, 128, 256 или 1024).

Запись предельных значений в регистр OCR1x связана с особыми случаями в генерации ШИМ- импульсов. Если OCR1x установить равным нижнему пределу (0x0000), то на выходе будет возникать короткий импульс каждый (ВП+1)-ый такт синхронизации таймера. Запись в OCR1x значения равного верхнему пределу приведет к установке постоянного уровня лог. 1 или 0 на выходе (зависит от выбранной с помощью бит COM1x1:0 полярности выходного сигнала).

Если требуется генерация меандра (прямоугольные импульсы со скважностью 2 или заполнением 50%) высокой частоты, то необходимо использовать режим быстрой ШИМ с установкой бит COM1A1:0 = 1, которая вызывает переключение (инвертирование) логического уровня на выходе OC1A при каждом совпадении. Данное применимо, только если OCR1A используется для задания верхнего предела (WGM13:0 = 15). Максимальная генерируемая частота меандра в этом случае fOC1A = fclk_I/O/2, если OCR1A =0x0000. Данная особенность аналогична переключению OC1A в режиме CTC за исключением двойной буферизации, которая имеется в режиме быстрой ШИМ.

Режим широтно-импульсной модуляции с фазовой коррекцией

Режим широтно-импульсной модуляции с фазовой коррекцией (ШИМ ФК) (WGM13:0 = 1, 2, 3, 10, или 11) предназначен для генерации ШИМ сигнала с фазовой коррекцией и высокой разрешающей способностью. Режим ШИМ ФК основан на двунаправленной работе таймера-счетчика. Счетчик циклически выполняет счет в направлении от нижнего предела (0x0000) до верхнего предела, а затем обратно от верхнего предела к нижнему пределу. Если задан неинвертирующий режим выхода формирователя импульсов, то выход OC1x сбрасывается/устанавливается при совпадении значений TCNT1 и OCR1x во время прямого/обратного счета. Если задан инвертирующий режим выхода, то, наоборот, во время прямого счета происходит установка, а во время обратного – сброс выхода OC1x. При двунаправленной работе максимальная частота ШИМ-сигнала меньше, чем при однонаправленной работе, однако, за счет такой особенности, как симметричность в режимах ШИМ с двунаправленной работой, данные режимы предпочитают использовать при решении задач управления приводами.

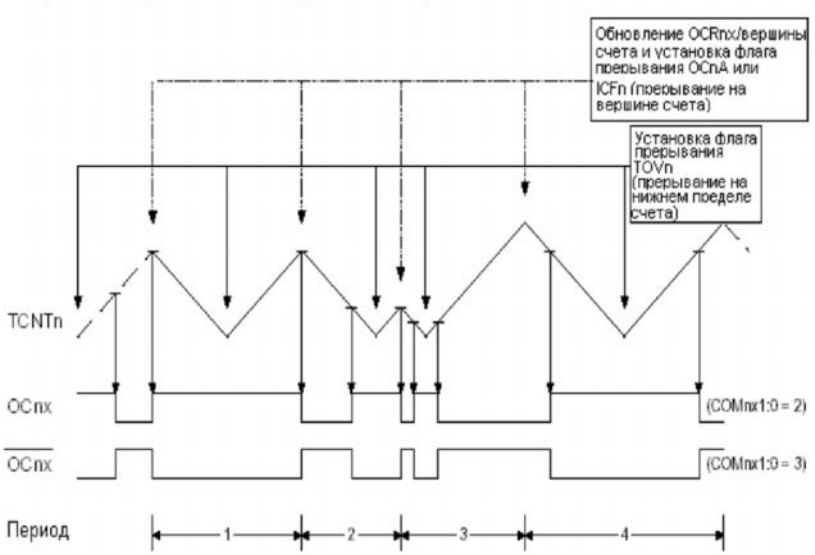
Разрешающая способность ШИМ в данном режиме может быть либо фиксированной (8, 9 или 10 разрядов) либо задаваться с помощью регистра ICR1 или OCR1A. Минимальная разрешающая способность равна 2-м разрядам (ICR1 или OCR1A =

0x0003), а максимальная -16-ти разрядам (ICR1 или OCR1A =0xFFFF). Если задан верхний предел, то разрешающая способность ШИМ в данном режиме определяется следующим образом:

$$R_{PCPWM} = \frac{\log(TOP+1)}{\log(2)}$$

В режиме ШИМ ФК счетчик инкрементируется пока не достигнет одного из фиксированных значений 0x00FF, 0x01FF или 0x03FF (соответственно для WGM13:0 = 1, 2, или 3), а также значения равного ICR1 (если WGM13:0 = 10) или OCR1A (если WGMn3:0 =11). Далее, при достижении верхнего предела, счетчик изменяет направление счета.Значение TCNT1 остается равным верхнему пределу в течение одного такта синхронизации таймера. Временная диаграмма для режима ШИМ ФК представлена на рисунке 39. На рисунке показан режим ШИМ ФК с использованием регистра OCR1A или ICR1 для задания верхнего предела. Состояние TCNT1 представлено в виде графика функции для иллюстрации двунаправленности счета. На рисунке представлены, как неинвертированный, так и инвертированный ШИМ-выход. Короткие горизонтальные линии указывают точки на графике изменения TCNT1, где возникает совпадение со значением OCR1x. Флаг прерывания OC1x устанавливается при возникновении совпадения.

Рисунок 39 Временная диаграмма для режима ШИМ ФК



Флаг переполнения таймера-счетчика (TOV1) устанавливается всякий раз, когда счетчик достигает нижнего предела. Если для задания верхнего предела используется регистр OCR1A или ICR1, то, соответственно устанавливается флаг OC1A или ICF1 тем же тактовым импульсом, на котором произошло обновление регистра OCR1x из буферного регистра (на вершине счета). Флаги прерывания могут использоваться для генерации прерывания по достижении счетчиком нижнего или верхнего предела.

При изменении значения верхнего предела счета необходимо следить, чтобы оно было больше или равно значениям во всех регистрах сравнения. В противном случае совпадение между TCNT1 и OCR1x никогда не возникнет. Обратите внимание, что при использовании фиксированных значений верхнего предела счета во время записи в регистры OCR1x неиспользуемые разряды обнуляются. Третий период на рисунке 39 иллюстрирует случай, когда динамическое изменение верхнего предела счета приводит к генерации несимметричного импульса. Данная особенность основывается на времени обновления регистра OCR1x. Поскольку, обновление OCR1x возникает на вершине счета, то и период ШИМ начинается и заканчивается на вершине счета. Это подразумевает, что длительность обратного счета определяется предыдущим значением верхнего предела, а прямого – новым значением верхнего предела.

Если два этих значения разные, то и длительность прямого и обратного счета будет также отличаться. Различие в длительности приводит несимметричности выходных импульсов.

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

В режиме ШИМ ФК блоки сравнения позволяют генерировать ШИМ-сигналы на выводах OC1x. Если установить COM1x1:0 = 2, то выход ШИМ будет без инверсии, а если COM1x1:0=3, то с инверсией (см. таблицу 38 странице 98). Фактическое значение OC1x можно наблюдать на выводе порта, если в регистре направления данных для данного вывода порта задано выходное направление (DDR_OC1x). ШИМ-сигнал генерируется путем установки (сброса) регистра OC1x при совпадении значений OCR1x и TCNT1 во время прямого счета, а также путем сброса (установки) регистра OC1x при совпадении между OCR1x и TCNT1 во время обратного счета. Результирующая частота ШИМ-сигнала в режиме ШИМ ФК при заданном верхнем пределе (ВП) может быть вычислена по следующему выражению:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

где N – коэффициент деления предделителя (1, 8, 32, 64, 128, 256 или 1024).

Запись предельных значений в регистр OCR1x связано с особыми случаями в генерации ШИМ- сигналов в режиме ШИМ ФК. Если задать режим ШИМ без инверсии и OCR1x установить равным нижнему пределу, то на выходе непрерывно будет установлен лог. 0, а если равным верхнему пределу, то на выходе постоянно присутствует лог. 1. Для ШИМ с инверсией указанные уровни необходимо заменить противоположными.

Если задать использование OC1A в качестве верхнего предела (WGM13:0 = 11) и установить COM1A1:0 =1, то на выходе OC1A будет генерироваться меандр.

Режим широтно-импульсной модуляции с фазовой и частотной коррекцией

Режим широтно-импульсной модуляции с фазовой и частотной коррекцией (ШИМ ФЧК) (WGM13:0 = 8 или 9) предназначен для генерации ШИМ-импульсов высокой разрешающей способности с фазовой и частотной коррекцией. Также как и режим ШИМ ФК режим ШИМ ФЧК основан на двунаправленной работе счетчика. Счетчик циклически считает от нижнего предела (0x0000) до верхнего предела, а затем обратно от верхнего предела к нижнему пределу. Если задан неинвертирующий режим ШИМ, то выход OC1x сбрасывается, если возникает совпадение между TCNT1 и OCR1x во время прямого счета, и устанавливается, если возникает совпадение во время обратного счета. В инвертирующем режиме работа инверсная. Двунаправленная работа, по сравнению с однонаправленной, связана с генерацией более низких частот. Однако, благодаря симметричности в режимах ШИМ с двунаправленным счетом, их применение предпочтительно в задачах управления приводами.

Основное отличие между режимами ШИМ ФК и ШИМ ФЧК состоит в моменте обновления регистра OCR1x из буферного регистра OCR1x (см. рисунок 39 и рисунок 40).

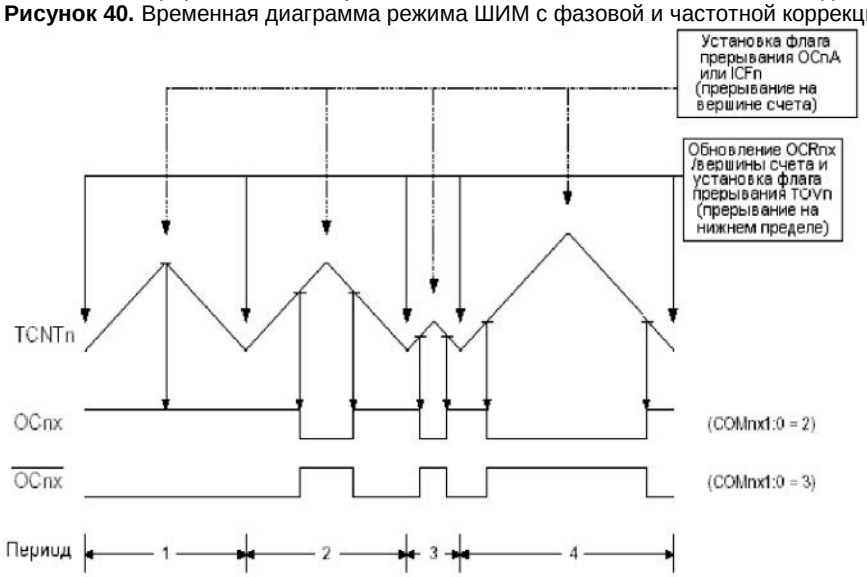
Разрешающая способность ШИМ в этом режиме может задаваться с помощью регистра ICR1 или OCR1A. Минимальная разрешающая способность равна 2-ум разрядам (ICR1 или OCR1A = 0x0003), а максимальная разрешающая способность - 16-ти разрядам (ICR1 или OCR1A = 0xFFFF). Разрешающая способность ШИМ в разрядах может быть вычислена по следующему выражению:

$$R_{PFCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

В режиме ШИМ ФЧК счетчик инкрементируется до совпадения со значением в ICR1 (WGM13:0 = 8) или в OCR1A (WGM13:0 = 9).

Это означает достижение вершины счета, после чего происходит изменение направления счета. Значение TCNT1 остается равным вершине счета в течение одного такта синхронизации таймера. Временная диаграмма для режима ШИМ ФЧК показана на рисунке 40. На рисунке показан режим ШИМ ФЧК, когда вершину счета задает регистр OCR1A или ICR1. Значение TCNT1 показано в виде графика функции для иллюстрации двунаправленности счета. На диаграмме показан как неинвертирующий, так и инвертирующий ШИМ выходы. Короткие горизонтальные линии указывают на точки график TCNT1, где возникает совпадение между OCR1x и TCNT1. Флаг прерывания OC1x устанавливается после возникновения совпадения.

Рисунок 40. Временная диаграмма режима ШИМ с фазовой и частотной коррекцией



Флаг переполнения таймера-счетчика (TOV1) устанавливается тем же тактом, когда произошло обновление регистров значением из буферного регистра (на нижнем пределе счета). Если для задания верхнего предела используется регистр OCR1A или ICR1, то по достижении счетчиком верхнего предела устанавливается флаг OC1A или ICF1, соответственно. Флаги прерывания могут использоваться для генерации прерывания при достижении счетчиком верхнего или нижнего предела.

При изменении верхнего предела необходимо следить, чтобы новое значение было больше или равно значениям во всех регистрах порога сравнения. В противном случае, если задано значение верхнего предела меньше любого из значений регистров порога сравнения, совпадение между TCNT1 и OCR1x никогда не наступит.

На рисунке 40 показано, что в отличие от режима ШИМ ФК, генерируемый выходной сигнал симметричен на всех периодах. Поскольку, регистры OCR1x обновляются на нижнем пределе счета, то длительности прямого и обратного счетов всегда равны. В результате выходные импульсы имеют симметричную форму, а, следовательно, и откорректированную частоту.

Использование регистра ICR1 для задания верхнего предела рекомендуется, если значение верхнего предела является константой. В этом случае также освобождается регистр OCR1A для широтно-импульсной модуляции импульсов на выводе OC1A. Однако если требуется динамическое изменение частоты ШИМ за счет изменения верхнего предела, то для задания верхнего предела рекомендуется использовать регистр OCR1A за счет наличия у него двойной буферизации.

В режиме ШИМ ФЧК блоки сравнения позволяют генерировать ШИМ-импульсы на выводе OC1x. Если COM1x1:0 = 2, то задается неинвертирующий ШИМ выход, а, если COM1x1:0=3, то инвертирующий (см. таблицу 38 стр.98).

Значение OC1x будет присутствовать на соответствующем выводе порта только в случае, если для него задано выходное направление(DDR_OC1x).. ШИМ сигнал генерируется путем установки (сброса) регистра OC1x при совпадении между OCR1x и TCNT1 во время прямого счета и сброса (установки) регистра OC1x при совпадении между OCR1x и TCNT1 во время обратного счета. Частота ШИМ в данном режиме при заданном верхнем пределе (ВП) счета определяется следующим образом:

$$f_{OCnxPFCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

где N – коэффициент деления предделителя (1, 8, 32, 64, 128, 256 или 1024).

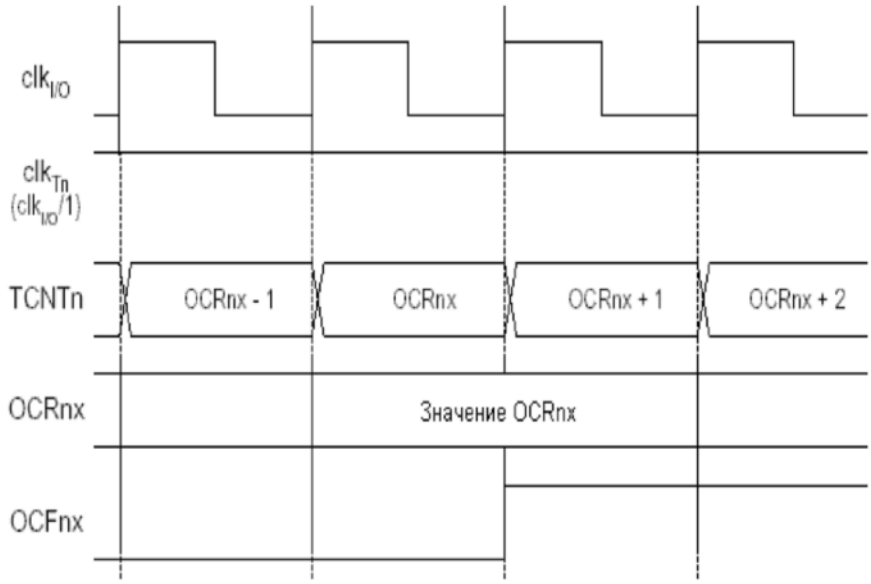
Запись предельных значений в регистр OCR1x связана с особыми случаями в генерации ШИМ- сигналов в данном режиме. Если задать OCR1x равным нижнему пределу (0x0000), то в неинвертирующем режиме на выходе будет постоянного присутствовать низкий логический уровень, а при записи значения равного верхнему пределу на выходе будет длительно присутствовать высокий логический уровень. В инвертирующем режиме приведенные уровни будут противоположными.

Если OCR1A используется для задания верхнего предела (WGM13:0 = 9) и COM1A1:0 = 1, то на выходе OC1A будет генерироваться меандр.

Временные диаграммы 16-разр. таймера-счетчика

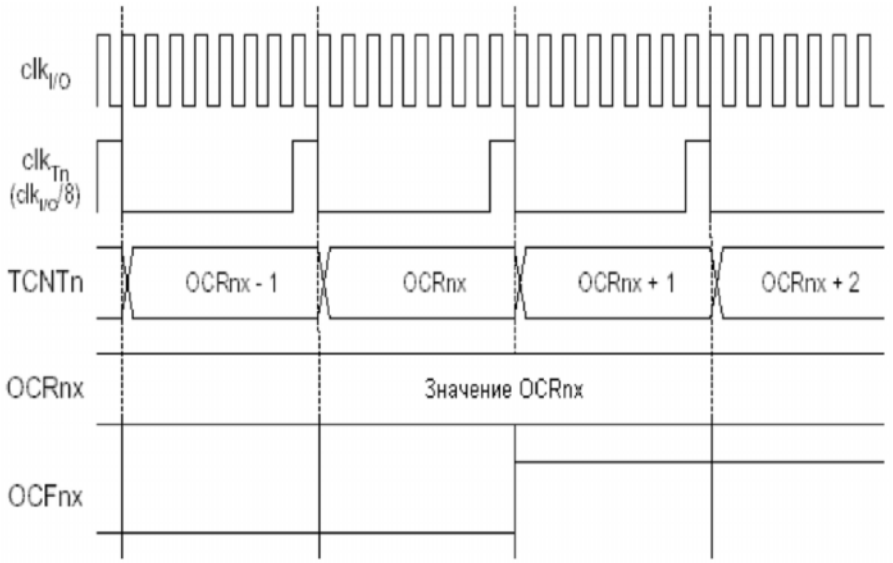
16-разр. таймер выполнен по синхронной схеме, поэтому, сигнал синхронизации таймера (clkT1) на следующих рисунках показан как разрешающий сигнал. На рисунках также представлена информация по моментам установки флагов прерываний и обновления регистра OCR1x значением из буферного регистра OCR1x (только для режимов с двойной буферизацией). На рисунке 41 представлена временная диаграмма с установкой флага OCF1x.

Рисунок 41. Временная диаграмма таймера-счетчика без предделения с установкой OCF1x



На рисунке 42 представлена аналогичная диаграмма, но с разрешенным предделением.

Рисунок 42. Временная диаграмма таймера-счетчика с предделением на 8 (fclk_I/O/8) и установкой OCF1x

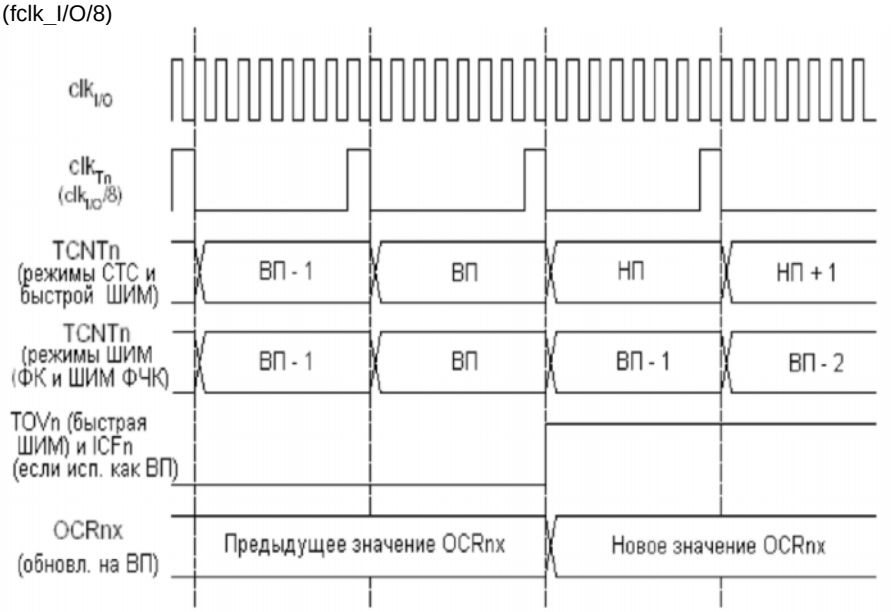


На рисунке 43 иллюстрируется алгоритм счета в районе верхнего предела в различных режимах. Если используется режим ШИМ ФЧК, то регистр OCR1x обновляется на нижнем пределе. В этом случае временная диаграмма будет такой же, но ВП необходимо заменить на НП, ВП-1 на НП+1 и т.д. Аналогичные переименования необходимо применить в режимах, где флаг TOV1 устанавливается на нижнем пределе.

Рисунок 43. Временная диаграмма таймера-счетчика без предделения

На рисунке 44 приведена аналогичная предыдущей временная диаграмма, но с разрешенным предделением.

Рисунок 44.Временная диаграмма таймера-счетчика с предделением на 8 (fclk_I/O/8)



Описание регистров 16-разр. таймера-счетчика

Регистр A управления таймером-счетчиком 1 – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• **Bit 7:6** – COM1A1:0: Режим формирования выходного сигнала канала A
• **Bit 5:4 –** COM1B1:0: Режим формирования выходного сигнала канала B

Биты COM1A1:0, COM1B1:0 влияют на работу выводов OC1A, OC1B и, соответственно. Если один или оба бита COM1A1:0 равны 1, то вывод OC1A переходит к выполнению альтернативной функции, запрещая его работу как обычного порта ввода-вывода. Аналогичные изменения происходят с выводом OC1B во время записи лог. 1 в один из битов COM1B1:0 . Однако необходимо учитывать, что остается влияние на работу данных выводов со стороны регистра направления данных (DDR) и в соответствующих разрядах этого регистра должно быть задано выходное направление для выводов OC1A, OC1B .

Если выбрано подключение сигналов OC1A, OC1B к выводам микроконтроллера, то назначение бит COM1x1:0 определяется выбранным с помощью бит WGM13:0 режима работы таймера-счетчика. В таблице 36 показано назначение бит COM1x1:0, когда битами WGM13:0 выбран режим сброса при совпадении (CTC) или нормальный режим, т.е. режимы без ШИМ.

Таблица 36. Режимы формирования выходного сигнала в режимах работы таймера без ШИМ

COM1A1/ COM1B1	COM1A0/ COM1B0	Описание
0	0	Нормальная работа порта, сигналы OC1A/OC1B отключены.
0	1	Переключение OC1A/OC1B при совпадении.
1	0	Сброс OC1A/OC1B при совпадении (установка лог. 0).
1	1	Установка OC1A/OC1B при совпадении (установка лог. 1).

В таблице 37 представлено назначение бит COM1x1:0, когда с помощью бит WGM13:0 выбран режим быстрой ШИМ.

Таблица 37. Сравните Выходной Режим, Быстрой ШИМ (1)

COM1A1/ COM1B1	COM1A0/ COM1B0	Описание
0	0	Нормальная работа порта, сигналы OC1A/OC1B отключены.
0	1	WGM13:0 = 15: Переключение OC1A при совпадении, OC1B отключен (нормальная работа порта). Для всех других установок WGM1 соответствует нормальная работа порта, когда OC1A/OC1B отключены.
1	0	Сброс OC1A/OC1B при совпадении, установка OC1A/OC1B на нижней вершине счета. (неинвертирование)
1	1	Установка OC1A/OC1B при совпадении, сброс OC1A/OC1B на нижней вершине. (инвертирование)

Прим.: Имеются особые случаи, когда OCR1A/OCR1B равно верхнему пределу счета и установлен COM1A1/COM1B1. В этом случае возникшее совпадение игнорируется, но установка или сброс на вершине счета выполняется (см. “Режим быстрой ШИМ”). на странице 89

В таблице 38 представлено назначение бит COM1x1:0, когда биты WGM13:0 установлены для режима ШИМ ФК и ШИМ ФЧК

Таблица 38. Режим формирования выходного сигнала в режимах работы таймера с ШИМ ФК и ШИМ ФЧК (1)

COM1A1/ COM1B1	COM1A0/ COM1B0	Описание
0	0	Нормальная работа порта, сигналы OC1A/OC1B отключены.
0	1	WGM13:0 = 9или 14: Переключение OC1A при совпадении, OC1B отключен (нормальная работа порта). Для всех других установок WGM1 соответствует нормальная работа порта, когда OC1A/OC1B отключены.
1	0	Сброс OC1A/OC1B при совпадении, во время прямого счета, установка OC1A/OC1B при совпадении во время обратного счета.
1	1	Установка OC1A/OC1B при совпадении, во время прямого счета сброс OC1A/OC1B при совпадении во время обратного счета.

Прим.: Имеются особые случаи, когда OCR1A/OCR1B равно верхнему пределу и установлен COM1A1/COM1B1 (см. “Режим ШИМ с фазовой коррекцией”). на странице 91

• **Bit 3 –** FOC1A: Режим формирования выходного сигнала канала A
• **Bit 2** – FOC1B :Режим формирования выходного сигнала канала B

Биты FOC1A/FOC1B являются активными только, когда биты WGM13:0 определяют режим без-ШИМ. Однако, для того, чтобы гарантировать совместимость будущими устройствами, эти биты должны быть установлены в 0, когда TCCR1A записывается, работая в режиме PWM. При записи логического 1 в биты FOC1A/FOC1B устанавливается режим сброса таймера при совпадении . Вывод OC1A/OC1B изменяется согласно его установке битов COM1x1:0. Отметьте, что биты FOC1A/FOC1B реализуются как стробы. Поэтому их значение и значение бит COM1x1:0, определяет, что включен режим CTC.

Строб FOC1A/FOC1B не будет генерировать прерывания, и при этом это не очистит таймер , используя OCR1A при очитке таймера на вершине счета в режиме CTC. Биты FOC1A/FOC1B всегда читаются как нуль.
• **Bit 1:0 –** WGM11:0: Режим работы таймера-счетчика

В сочетании с битами WGM13:2 из регистра TCCR1B данные биты определяют алгоритм счета, источник для задания вершины счета (ВП) и тип генерируемой формы сигнала (см. табл. 39).Таймер-счетчик может работать в одном из следующих режимов: нормальный режим (счетчик), сброс таймера при совпадении (СТС) и три режима с широтно-импульсной модуляцией (ШИМ) (см. "Режимы работы 16-разр. таймеров-счетчиков " на странице 88).

Таблица 39. Режимы работы таймера-счетчика

Режим	**WGM13**	**WGM12 (CTC1)**	**WGM11 (PWM11)**	**WGM10 (PWM10)**	**Режим работы таймера-счетчика(1)**	**Верхний предел**	**Обновление OCR1x**	**Установка флага TOV1 на:**
0	0	0	0	0	Нормальный	0xFFFF	сразу после записи	МАКС
1	0	0	0	1	8-разр. ШИМ ФК	0x00FF	на вершине счета	нижнем пределе
2	0	0	1	0	9-разр. ШИМ ФК	0x01FF	на вершине счета	нижнем пределе
3	0	0	1	1	10-разр. ШИМ ФК	0x03FF	на вершине счета	нижнем пределе
4	0	1	0	0	СТС	OCR1A	сразу после записи	МАКС
5	0	1	0	1	8-разр.быстрая ШИМ	0x00FF	нижнем пределе	на вершине счета
6	0	1	1	0	9-разр.быстрая ШИМ	0x01FF	нижнем пределе	на вершине счета
7	0	1	1	1	10-разр.быстрая ШИМ	0x03FF	нижнем пределе	на вершине счета
8	1	0	0	0	ШИМ ФЧК	ICR1	нижнем пределе	нижнем пределе
9	1	0	0	1	ШИМ ФЧК	OCR1A	нижнем пределе	нижнем пределе
10	1	0	1	0	ШИМ ФК	ICR1	на вершине счета	нижнем пределе
11	1	0	1	1	ШИМ ФК	OCR1A	на вершине счета	нижнем пределе
12	1	1	0	0	СТС	ICR1	сразу после записи	МАКС
13	1	1	0	1	(резерв)	–	–	–
14	1	1	1	0	Быстрая ШИМ	ICR1	нижнем пределе	на вершине счета
15	1	1	1	1	Быстрая ШИМ	OCR1A	нижнем пределе	на вершине счета

Прим.: 1. Наименования бит CTC1 и PWM11:0 являются устаревшими, поэтому, необходимо использовать имена WGM12:0. Однако назначение и расположение этих бит совместимо с предыдущими версиями таймеров.

Регистр B управления таймером-счетчиком 1 – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• **Bit 7 – ICNC1:** Подавитель шума на входе захвата
Установка данного бита (запись лог. 1) активизирует подавитель шума на входе захвата. После активизации подавителя шумов сигнал с вывода ICP1 пропускается через фильтр. Логика работы фильтра состоит в определении четырех подряд равных по значению выборок и только в этом случае изменении своего выходного состояния. Следовательно, после разрешения подавления шумов сигнал с входа захвата будет задерживаться на 4 такта системной синхронизации.

• **Bit 6 – ICES1:** Выбор детектируемого фронта на входе захвата
Данный бит позволяет задать, какой фронт на входе захвата ICP1 приведет к захвату состояния таймера. Если ICES1 =0, то падающий (отрицательный) фронт приводит к захвату состояния таймера, а если же ICES1 = 1, то нарастающий (положительный) фронт приводит к возникновению захвата.

Если в соответствии с установкой ICES1 возникает условие захвата, то содержимое счетчика копируется в регистр захвата ICR1. При этом также устанавливается флаг захвата ICF1, который может использоваться для генерации прерывания по захвату (если данное прерывание разрешено).

Если регистр ICR1 используется для хранения значения верхнего предела счета (см. описание битов WGM13:0, расположенных в TCCR1A и Регистре TCCR1B), то вход ICP1 отключается от соответствующего вывода микроконтроллера и функция захвата блокируется.

• **Bit 5** – Зарезервированный бит
Данный бит зарезервирован для дальнейшего использования. В целях совместимости с будущими разработками рекомендуется во время записи в регистр TCCR1B в данном разряде указывать лог. 0.

• **Bit 4:3 – WGM13:2:** Режим работы таймера-счетчика
См. описание регистр TCCR1A.

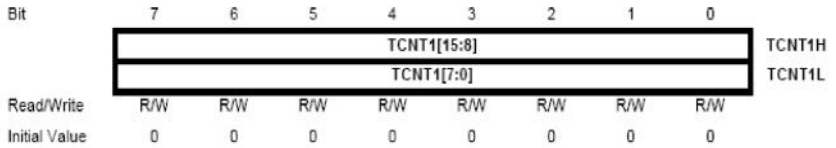
• **Bit 2:0 – CS12:0**: Выбор тактового источника
Данные три бита позволяют выбрать тактовый источник для таймера-счетчика (см. рисунок 41 и рисунок 42).

Таблица 40. Описание бит выбора тактового источника

CSn2	CSn1	CSn0	Описание
0	0	0	Нет синхронизации. Таймер-счетчик остановлен.
0	0	1	clkI/O/1 (без предделения)
0	1	0	clkI/O /8 (с предделением)
0	1	1	clkI/O/64 (с предделением)
1	0	0	clkI/O/256 (с предделением)
1	0	1	clkI/O/1024 (с предделением)
1	1	0	Внешний тактовый источник с выв. Tn. Синхронизация по падающему фронту.
1	1	1	Внешний тактовый источник с выв. Tn. Синхронизация по нарастающему фронту.

Если для тактирования таймера выбран внешний вывод T1, то данная функция за ним сохраняется, даже при его настройке на вывод. Данная функция позволяет программно управлять счетом.

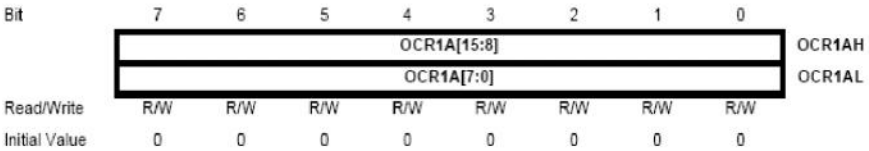
Таймер-счетчик 1 – TCNT1H и TCNT1L



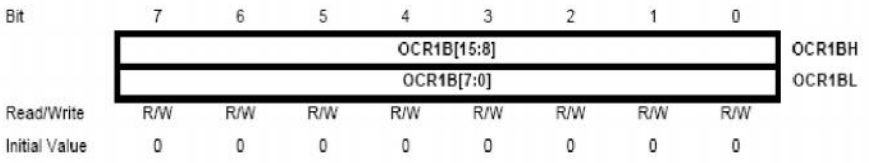
Две ячейки в области ввода-вывода (TCNT1H и TCNT1L, вместе TCNT1) дают полный доступ, как на чтение, так и на запись к 16-разрядному счетчику. В целях гарантирования одновременности чтения и записи старшего и младшего байтов этих регистров, доступ организован с использованием 8-разрядного временного регистра старшего байта (TEMP). Временный регистр является общим для всех 16-разрядных регистров таймера (см. также “Доступ к 16-разр. регистрам”). на странице 79

Изменение содержимого счетчика TCNT1 во время его работы (счета) связано с риском возникновения совпадения между TCNT1 и одним из регистров OCR1x. Запись в регистр TCNT1 блокирует отработку совпадения, которое возникнет на следующем такте, для всех блоков сравнения.

Регистр сравнения 1 A – OCR1AH и OCR1AL



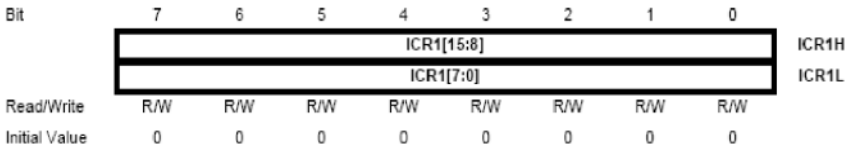
Регистр сравнения 1 B – OCR1BH и OCR1BL



В регистрах сравнения хранится 16-разр. значение, которое непрерывно сравнивается со значением счетчика (TCNT1). Возникающее совпадение может использоваться для генерации прерывания по результату сравнения и генерации прямоугольных импульсов на выводе OC1x.

Регистры сравнения являются 16-разрядными, поэтому, одновременность записи младшего и старшего байтов достигнута за счет использования 8-разр. временного регистра старшего байта (TEMP). Временный регистр является общим для всех 16-разрядных регистров таймера (см. также “Доступ к 16-разр. регистрам”). на странице 79

Регистр захвата 1 – ICR1H и ICR1L

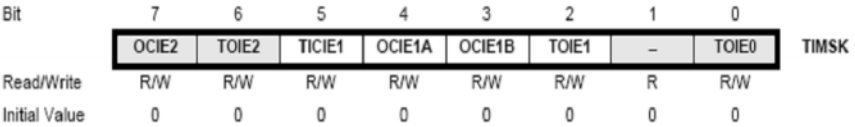


Регистры захвата обновляются содержимым соответствующего счетчика (TCNT1) при каждом определении условия захвата на входе ICP1 (или альтернативно на выходе аналогового компаратора для таймера-счетчика 1).

Регистры захвата альтернативно могут использоваться для задания верхнего предела счета.

Регистры захвата также являются 16-разрядными, поэтому, одновременность записи младшего и старшего байтов достигнута за счет использования 8-разр. временного регистра старшего байта (TEMP). Временный регистр является общим для всех 16-разрядных регистров таймера (см. также “Доступ к 16-разр. регистрам”). странице 79.

Регистр маски прерываний таймера-счетчика – TIMSK



Прим.: Данный регистр биты управления прерываниями для нескольких таймер-счетчиков, но в данном разделе детализированы только биты таймера 1. Описание остальных бит необходимо искать при описании соответствующих таймеров.

• **Bit 5 – TICIE1:** Разрешение прерывания по захвату состояния таймера-счетчика 1
Если в данный бит записана лог. 1, а также установлен флаг I в регистре статуса (активно общее разрешение прерываний), то разрешается прерывание по захвату состояния таймера- счетчика 1. Если устанавливается флаг ICF1в регистре TIFR, программа переходит на соответствующий вектор прерывания (см. раздел "Прерывания"). на странице 46

• **Bit 4 – OCIE1A:** Разрешение прерывания по результату сравнения канала A таймера- счетчика 1
Если в данный бит записана лог. 1 и установлен флаг I в регистре статуса, то разрешается работа прерывания по результату сравнения канала A. Если устанавливается флаг OCF1A в регистре TIFR, то программа переходит на соответствующий вектор прерываний (см. раздел "Прерывания"). на странице 46)

• **Bit 3 – OCIE1B:** Разрешение прерывания по результату сравнения канала B таймера- счетчика 1
Если в данный бит записана лог. 1 и установлен флаг I в регистре статуса, то разрешается работа прерывания по результату сравнения канала A. Если устанавливается флаг OCF1B в регистре TIFR, то программа переходит на соответствующий вектор прерываний (см. раздел "Прерывания"). на странице 46)

• **Bit 2 – TOIE1:** Разрешение прерывания при переполнении таймера-счетчика 1
Если в данный бит записана лог. 1 и установлен флаг I в регистре статуса, то разрешается прерывание по переполнению таймера-счетчика 1. После этого, установка флага TOV1 в регистре TIFR приведет к переходу на соответствующий вектор прерывания.

Регистр флагов прерываний таймеров-счетчиков – TIFR

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Прим.: Биты данного регистра относятся к нескольким таймерам, но в данном параграфе рассматриваются биты только одного таймера. Описание остальных бит необходимо смотреть в соответствующих разделах.

• **Bit 5 – ICF1**: Флаг захвата состояния таймера-счетчика 1
Флаг устанавливается, если на входе ICP1 определяется условие захвата. Если регистр захвата ICR1 выбран с помощью бит WGM13:0 в качестве источника верхнего предела счета, флаг ICF1 устанавливается по достижении верхнего предела счета.

ICF1 автоматически сбрасывается при переходе на вектор прерывания по захвату состояния таймера-счетчика. Альтернативно флаг ICF1 можно сбрасывать путем записи в него лог. 1.

• **Bit 4 – OCF1A:** Флаг результата сравнения канала A таймера-счетчика 1
Данный флаг устанавливается следующим тактом после совпадения значения TCNT1 с регистром A порога сравнения (OCR1A). Обратите внимание, что строб принудительной установки результата сравнения (FOC1A) не устанавливает флаг OCF1A. Флаг OCF1A автоматически сбрасывается при переходе на соответствующий вектор прерывания. Альтернативно, флаг OCF1A сбрасывается путем записи в него лог. 1.

• **Bit 3 – OCF1B:** Флаг результата сравнения канала B таймера-счетчика 1
Данный флаг устанавливается следующим тактом после совпадения значения TCNT1 с регистром B порога сравнения (OCR1B). Обратите внимание, что строб принудительной установки результата сравнения (FOC1B) не устанавливает флаг OCF1B. Флаг OCF1B автоматически сбрасывается при переходе на соответствующий вектор прерывания. Альтернативно, флаг OCF1B сбрасывается путем записи в него лог. 1.

• **Bit 2 – TOV1:** Флаг переполнения таймера-счетчика 1
Установка данного флага зависит от значений бит WGM13:0. В нормальном режиме и режиме CTC флаг TOV1 устанавливается при переполнении таймера-счетчика. См. табл. 39 для изучения поведения флага TOV1 при задании других значений WGM13:0. Флаг TOV1 автоматически сбрасывается при переходе на вектор прерывания по переполнению таймера-счетчика 1. Альтернативно флаг TOV1 сбрасывается путем записи в него лог. 1.

8-разр. таймер-счетчик 2 с функциями широтно-импульсной модуляции и асинхронного тактирования

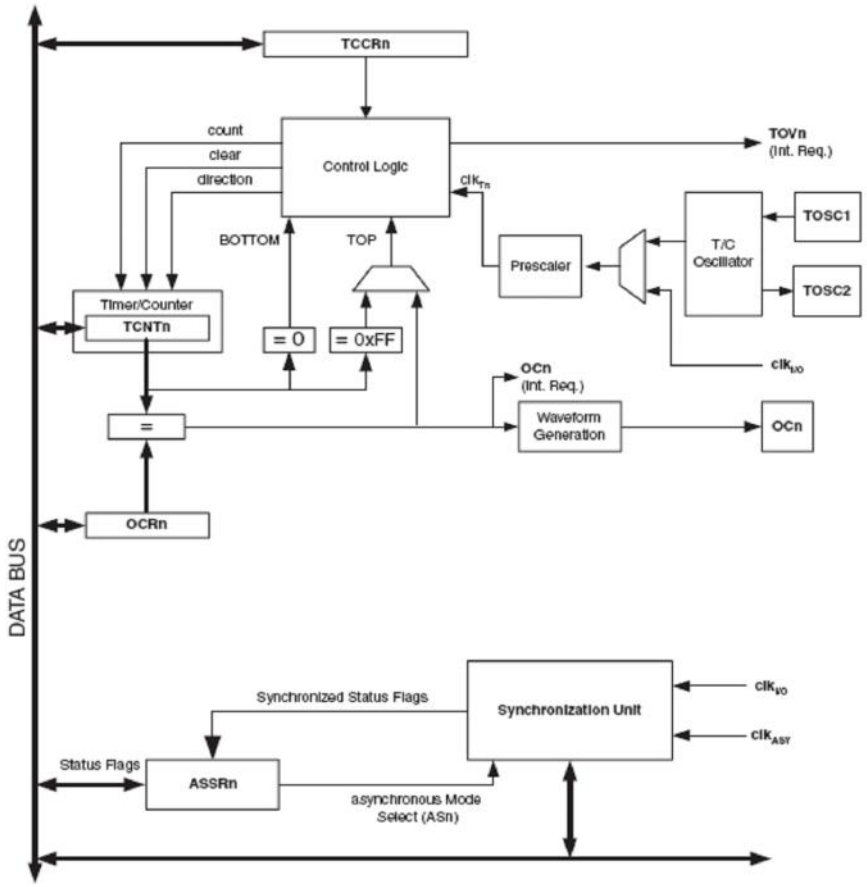
Таймер-счетчик 2- модуль многофункционального одноканального 8-разрядного таймера-счетчика. Основные функции:

- **Одноканальный счетчик**
- **Опциональный режим сброса таймера при совпадении (автоматическая перезагрузка)**
- **Широтно-импульсная модуляция без генерации ложных импульсов с фазовой коррекцией**
- **Генератор частоты**
- **10-разрядный предделитель тактовой частоты**
- **Генерация прерываний по переполнению и выполнения условия сравнения (TOV2 и OCF2)**
- **Возможность асинхронного тактирования совместно с внешним кварцевым резонатором частотой 32 кГц независимо от частоты синхронизации ввода-вывода**

Введение

Укрупненная функциональная схема 8-разр. таймера-счетчика представлена на рис. 45. Для уточнения расположения выводов см. "Расположение выводов". Связи с регистрами, к которым осуществляет доступ ЦПУ, в т.ч. биты ввода-вывода и линии ввода-вывода показаны жирной линией. Специфические для данного устройства регистры, расположение и назначение его бит приведены в "Описание регистров 8-разр. таймера-счетчика 2" страница 117.

Рисунок 45. Функциональная схема 8-разр. таймера-счетчика



Регистры

Регистр таймера-счетчика (TCNT2) и регистр порога сравнения (OCR2) - 8-разр. регистры. Сигналы запроса на прерывание представлены как флаги прерываний таймера в регистре TIFR. Все прерывания индивидуально маскируются с помощью регистра маски прерываний таймеров (TIMSK). Регистры TIFR и TIMSK не представлены на функциональной схеме, т.к. они совместно используются с другими таймерами микроконтроллера.

Таймер-счетчик может тактироваться через предделитель внутренне или асинхронно через внешние выводы TOSC1/2, что описано в последующих разделах. Асинхронная работа управляется регистром асинхронного состояния (ASSR). Блок синхронизации осуществляет выбор, какой тактовый источник используется для инкрементирования (декрементирования) состояния таймера-счетчика. Если источник тактирования не задан, то таймер-счетчик находится в неактивном состоянии. Выход логики выбора синхронизации обозначен как синхронизация таймера (clkT2).

Значение регистра порога сравнения с двойной буферизацией (OCR2) непрерывно сравнивается со значением таймера-счетчика. Результат сравнения может использоваться для генерации сигналов с ШИМ или прямоугольных импульсов переменной частоты на выводе OC2. См. "Блок сравнения" на странице 107. Совпадение порога сравнения со значением таймера-счетчика приводит к установке флага результата сравнения (OCF2), который может использоваться для генерации запроса на прерывание по результату сравнения.

Определения

Многие регистры, и разрядные ссылки в этом документе пишутся в общей форме. В более общем случае “n” заменяет число Таймера/Счетчика, в этом случае 2. Однако, при использовании регистра или бита в программе, должна использоваться точная форма записи – TCNT2 для того, чтобы получить значение Таймера/Счетчика0. Некоторые определения и их сокращенные наименования, которые интенсивно используются в этом разделе, представлены в таблице 41.

Таблица 41. Определения

НП (нижний предел)	Счетчик достигает нулевого значения (0x00)
МАКС (максимальное значение)	Счетчик достигает максимального значения 0xFF (десятич. 255)
ВП (верхний предел)	Счетчик достигает верхнего предела счета (вершина счета). В качестве вершины счета может выступать фиксированное значение 0xFF или содержимое регистра OCR2

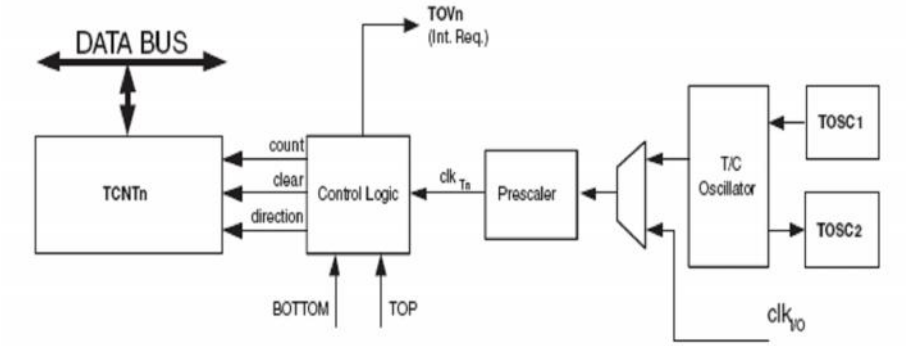
Тактовые источники таймера-счетчика 2

Таймер-счетчик 2 может тактироваться внутренне синхронно или внешне асинхронно (по отношению к внутренней системной синхронизации). По умолчанию используется тактовый сигнал clkT2, эквивалентный тактовому сигналу микроконтроллера clkI/O. Если в бит AS2 регистра ASSR записать лог. 1, то в качестве источника синхронизации выступает генератор таймера-счетчика, связанный с выводами подключения низкочастотного кварцевого резонатора TOSC1 и TOSC2. Более подробно асинхронная работа описана в "Регистр асинхронного состояния - ASSR" на странице 119. Подробности по источникам синхронизации см. в разделе "Предделитель таймера-счетчика 2" странице 123.

Блок счетчика

Основу 8-разр. таймера-счетчика 2 составляет программируемый двунаправленный счетчик. Рисунок 46 показывает функциональную схему счетчика и окружающих его элементов.

Рисунок 46. Функциональная схема счетчика



Описание сигналов (внутренние сигналы):

Count(Счет) - Инкрементирует или декрементирует TCNT2 на 1.
Direction(Направление) - Задает направление счета: инкрементирование (+1, прямой счет) или декрементирование (-1, обратный счет).
Clear(Сброс) - Сбрасывает содержимое TCNT2 (запись лог. 0 во все разряды).
clkT2 - Синхронизация таймера-счетчика.
TOP(Верхний предел) - Задает максимальное значение, которое может достигнуть TCNT2.
BOTTOM(Нижний предел) - Задает минимальное значение, которое может достигнуть TCNT2 (ноль).

В зависимости от выбранного режима работы счетчик сбрасывается, инкрементируется или декрементируется на каждом такте синхронизации (clkT2). Тактовый сигнал clkT2 может быть внутренним или внешним, а его частота выбирается с помощью бит выбора частоты синхронизации (CS22:0). Если источник синхронизации не задан (CS22:0=0), то таймер останавливается. Однако состояние TCNT2 доступно ЦПУ независимо от того работает синхронизация таймера или нет. Запись в регистр таймера через ЦПУ перекрывает любые действия самого счетчика: сброс или счет, т.е. имеет более высокий приоритет.

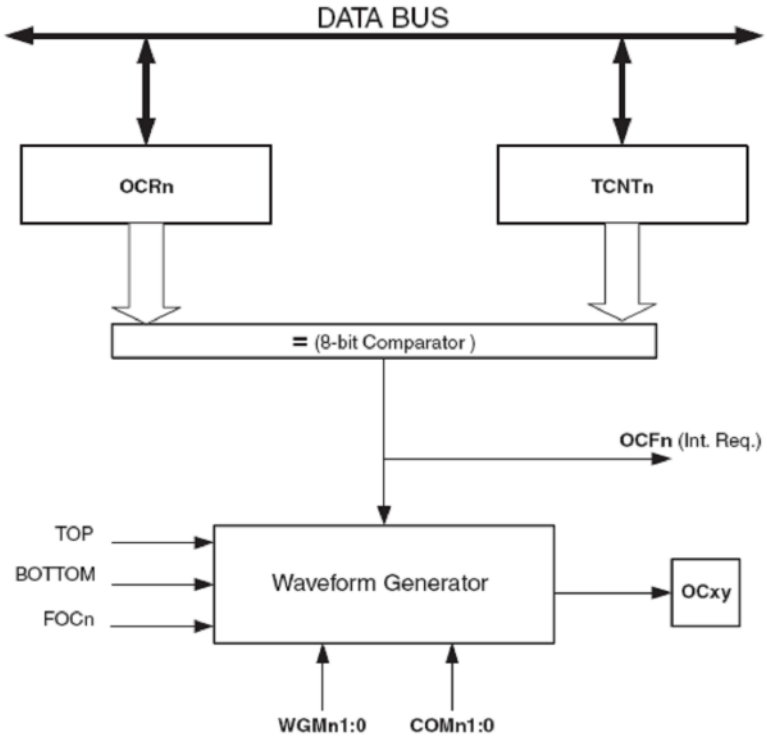
Последовательность счета определяется установкой бит WGM21 и WGM20, расположенных в регистре управления таймером-счетчиком (TCCR2). Имеется точная связь между поведением счетчика (алгоритмом счета) и генерируемой на выходе OC2 формы сигнала. Более подробно об алгоритмах счета и генерации импульсов написано в "Режимы работы" на странице 110.

Флаг переполнения таймера-счетчика (TOV2) устанавливается в соответствии с режимом работы, который выбирается битами WGM21, WGM20. Бит TOV2 может использоваться для генерации прерывания ЦПУ.

Блок сравнения

8-разрядный цифровой компаратор непрерывно выполняет сравнение содержимого регистра таймера-счетчика TCNT2 с регистром порога сравнения OCR2. Всякий раз, когда значение TCNT2 совпадает со значением OCR2 компаратор устанавливает флаг совпадения OCF2 следующим тактом синхронизации таймера. Если разрешено прерывание битом OCIE2 = 1, то установка флага совпадения вызывает запрос на прерывание. Флаг OCF2 автоматически сбрасывается во время выполнения процедуры обработки прерывания. Альтернативно, флаг OCF2 можно сбросить программно путем записи лог. 1 в позицию данного бита. Генератор сигнала использует сигнал результата сравнения для генерации прямоугольных импульсов по одному из алгоритмов, который выбирается битами задания режима работы таймера WGM21, WGM20 и битами задания режима формирования выходного сигнала (COM21, COM20). Верхний и нижний пределы счета используются в некоторых режимах работы для выполнения специальных действий (см. "Режимы работы таймера -счетчика 2") на странице 110 . На рисунке 47 приведена функциональная схема блока сравнения.

Рисунок 47. Функциональная схема блока сравнения



Регистр OCR2 выполнен по схеме двойной буферизации при использовании режимов с широтно-импульсной модуляцией (ШИМ). В нормальном режиме и режиме сброса таймера при совпадении (CTC) схема двойной буферизации отключается. Двойная буферизация позволяет синхронизировать обновление регистра сравнения OCR2 по достижении верхнего или нижнего предела счета. Такая синхронизация предотвращает возможность возникновения несимметричных ШИМ-импульсов нечетной длины, тем самым гарантируя отсутствие сбоев при генерации прямоугольных импульсов.

Доступ к регистру OCR2 может показаться сложным, но это выполнено не случайно. После разрешения двойной буферизации ЦПУ осуществляет доступ к буферному регистру OCR2, а после отключения - непосредственно адресуется к регистру OCR2.

Принудительная установка результата сравнения

В режимах генерации импульсов без ШИМ в формирователе импульсов результат сравнения может быть установлен непосредственно через бит принудительной установки результата сравнения FOC2. Принудительная установка результата сравнения компаратора не приводит к установке флага OCF2 или сбросу/перезагрузке таймера, но влияет на состояние вывода OC2, который будет устанавливаться, сбрасываться или переключаться (инвертироваться) в зависимости от выбранной установки бит COM21, COM20.

Результат сравнения блокируется записью в TCNT2

Если ЦПУ осуществляет запись в регистр TCNT2, то результат сравнения будет игнорироваться на следующем такте синхронизации таймера, даже если таймер остановлен. Данная функция позволяет установить в регистре OCR2 то же значение, что и в TCNT2 без генерации запроса на прерывание, если разрешено тактирование таймера-счетчика.

Использование блока сравнения

Поскольку запись в TCNT2 блокирует любые действия по результату сравнения на один такт синхронизации таймера независимо от режима работы, то при изменении TCNT2 при использовании канала сравнения (независимо работает синхронизация таймера или нет) необходимо учесть следующие особенности. Если в регистр TCNT2 записано значение равное OCR2, то игнорирование совпадения приведет к генерации некорректной формы сигнала. По аналогии следует избегать записи в TCNT2 значения равного нижнему пределу (0x00), если счетчик работает как вычитающий.

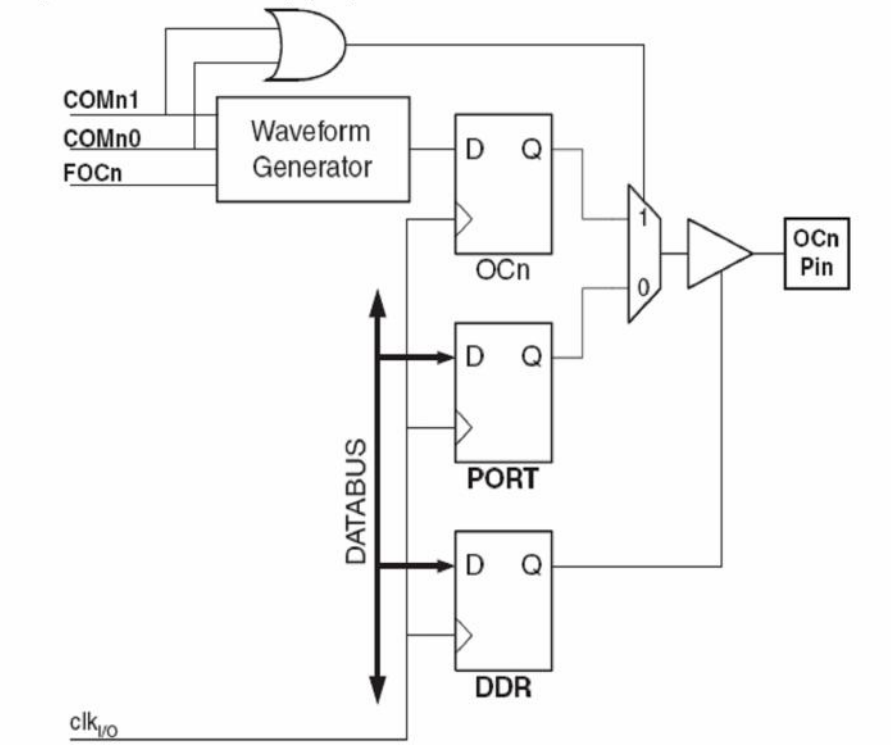
Установка OC2 должна быть выполнена перед настройкой линии ввода-вывода на вывод в регистре направления данных. Самый легкий путь установки значения OC2 - использование бита принудительной установки результата сравнения (FOC2) в нормальном режиме. Регистр OC2 сохраняет его значение, даже если происходит изменение режима работы таймера.

Учтите, что биты COM21, COM20 не содержат схемы двойной буферизации и на любые изменения реагируют мгновенно.

Блок формирования выходного сигнала

Биты задания режима формирования выходного сигнала (COM21:0) имеют двойное назначение. С одной стороны биты COM21, COM20 используются формирователем сигнала и определяют какое логическое состояние должно быть на выходе OC2 при возникновении следующего совпадения. С другой стороны, биты COM21, COM20 используются для разрешения/запрета альтернативной функции вывода порта OC2. На рисунке 48 представлена упрощенная логическая схема, на которую воздействуют биты COM21, COM20. На рисунке показаны только те регистры управления портом ввода-вывода (DDR и PORT), на которые оказывает действие биты COM21, COM20.

Рисунок 48. Схема блока формирования выходного сигнала



Функция линии универсального порта ввода-вывода заменяется на функцию выхода формирователя сигнала OC2, если хотя бы один из бит COM21, COM20 установлен (логика ИЛИ). Однако, управлением направлением вывода OC2 (вход или выход) в этом случае остается за соответствующим битом регистра направления данных порта B (DDRB.3). Чтобы значение регистра OC2 присутствовало на выводе OC2 необходимо настроить данную линию на вывод (установить бит DDRB.3). Управление вводом альтернативной функции не зависит от режима генерации сигнала.

Схемотехника выходной логики позволяет инициализировать состояние регистра OC2 перед разрешением настройки вывода OC2 в качестве выхода. Обратите внимание, что в некоторых режимах работах имеются зарезервированные состояния бит COM21, COM20. См. "Описание регистров 8-разр. таймера-счетчика 2" на странице 117.

Режимы генерации сигнала

Значение бит COM21:0 задает различные режимы формирования выходного сигнала, которые в свою очередь зависят от выбранного режима работы таймера. Таймер-счетчик 2 может быть переведен в нормальный режим, в режим сброса таймера при совпадении или в один из режимов с генерацией ШИМ-сигналов. Общим для всех режимов является невыполнение каких-либо действий с выводом OC2 при выполнении условия сравнения, если оба бита COM21, COM20 равны нулю. В таблице 43 на странице 118 описано действие различных установок этих бит для режимов без ШИМ. Аналогичная информация для режима с быстрой ШИМ приведена в таблице 44 на странице 118, а для ШИМ с фазовой коррекцией в таблице 45 странице 118.

После установки бит COM21, COM20 они вступают в силу только после первого возникшего совпадения вслед за этой установкой. Для режимов без ШИМ установки могут быть незамедлительно активизированы с помощью стробирующего бита FOC2.

Режимы работы таймера-счетчика 2

Режим работы таймера, в т.ч. поведение таймера-счетчика и связанного с ним выхода формирователя сигнала, задается комбинацией бит, задающих режим работы таймера (WGM21, WGM20) и режим формирования выходного сигнала (COM21, COM20). При этом биты задания режима формирования выходного сигнала не влияют на алгоритм счета, т.к. алгоритм счета зависит только от состояния бит задания режима работа таймера. В режимах с ШИМ биты COM21, COM20 позволяют включить/отключить инверсию на генерируемом ШИМ-выходе (т.е. выбрать ШИМ с инверсией или ШИМ без инверсии). Для режимов без ШИМ биты COM21:0 определяют какое действие необходимо выполнить при выполнении условия сравнения: сбросить, установить или инвертировать выход (см. также "Блок формирования выходного сигнала" на странице 109).

Более подробная информация по временным диаграммам работы таймера-счетчика 2 приведена в разделе "Временные диаграммы таймера-счетчика 2" на странице 115.

Нормальный режим работы

Самым простым режимом работы является нормальный режим (WGM21:0 = 0). В данном режиме счетчик работает как суммирующий (инкрементирующий), при этом сброс счетчика не выполняется. Переполнение счетчика происходит при переходе через максимальное 8-разр. значение (верхний предел = 0xFF) к нижнему пределу счета (0x00). В нормальном режиме работы флаг переполнения таймера-счетчика TOV2 будет установлен на том же такте синхронизации, когда TCNT2 примет нулевое значение. Фактически, флаг переполнения TOV2 является 9-ым битом таймера-счетчика за тем исключением, что он только устанавливается и не сбрасывается. Однако программно это свойство может быть использовано для повышения разрешающей способности таймера, если использовать прерывание по переполнению таймера, при возникновении которого флаг TOV2 сбрасывается автоматически. Для нормального режима работы не существует каких-либо особых ситуаций связанных с записью нового состояния счетчика, когда требовалось бы учесть меры предосторожности.

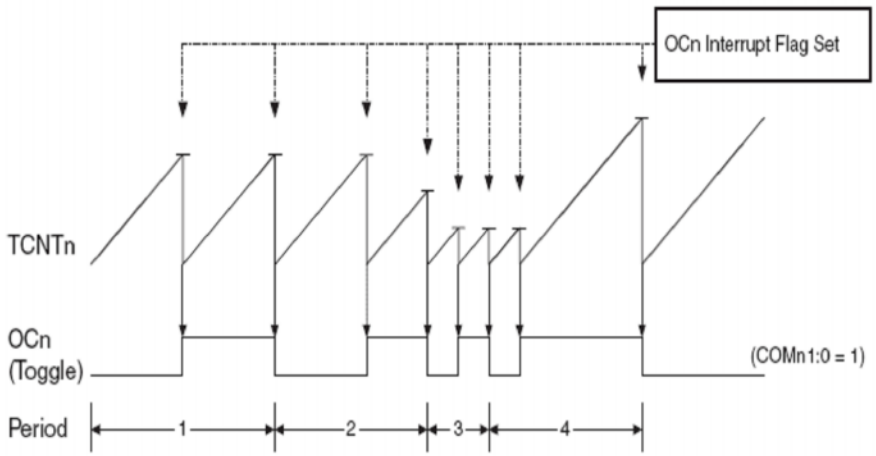
Блок сравнения может использоваться для генерации прерываний. Не рекомендуется использовать выход OC2 для генерации сигналов в нормальном режиме работы, т.к. в этом случае будет затрачена значительная часть процессорного времени.

Режим сброса таймера при совпадении (CTC)

В режиме CTC (WGM21, WGM20 =0b10) регистр OCR2 используется для задания разрешающей способности счетчика. Если задан режим CTC и значение счетчика (TCNT2) совпадает со значением регистра OCR2, то счетчик обнуляется (TCNT2=0). Таким образом, OCR2 задает вершину счета счетчика, а, следовательно, и его разрешающую способность. В данном режиме обеспечивается более широкий диапазон регулировки частоты генерируемых прямоугольных импульсов. Он также упрощает работу счетчика внешних событий.

Временная диаграмма для режима CTC показана на рисунке 49. Значение счетчика (TCNT2) инкрементируется до тех пор, пока оно не станет равным значению в OCR2, после чего счетчик (TCNT2) обнулится.

Рисунок 49. Временная диаграмма для режима CTC



С помощью флага OCF2 прерывание может генерироваться всякий раз, когда счетчик достигает своего верхнего предела счета. Если работа прерывания разрешена, то процедура обработки прерывания может использоваться для обновления значения вершины счета. Однако, задание значения вершины счета близкого к значению нижнего предела счета, когда счетчик работает без предделения или с малым значением предделения, необходимо выполнять с особой осторожностью, т.к. в режиме CTC нет двойной буферизации. Если значение, записанное в OCR2, меньше текущего значения TCNT2, то сброс счетчика по условию совпадения наступит, когда он достигнет максимального значения (0xFF), затем перейдет в исходное состояние 0x00 и достигнет нового значения OCR2.

Для генерации сигнала в режиме CTC выход компаратора OC2 может использоваться для изменения логического уровня при каждом совпадении, для чего необходимо задать режим переключения (COM21, COM20 = 0b01). Значение OC2 будет присутствовать на выводе порта, только если для данного вывода задано выходное направление. Максимальная частота генерируемого сигнала равна fOC0 = fclk_I/O/2, если OCR2 = 0x00. Для других значений OCR2 частоту генерируемого сигнала можно определить по формуле:

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

где переменная N задает коэффициент деления предделителя (1, 8, 32, 64, 128, 256 или 1024).

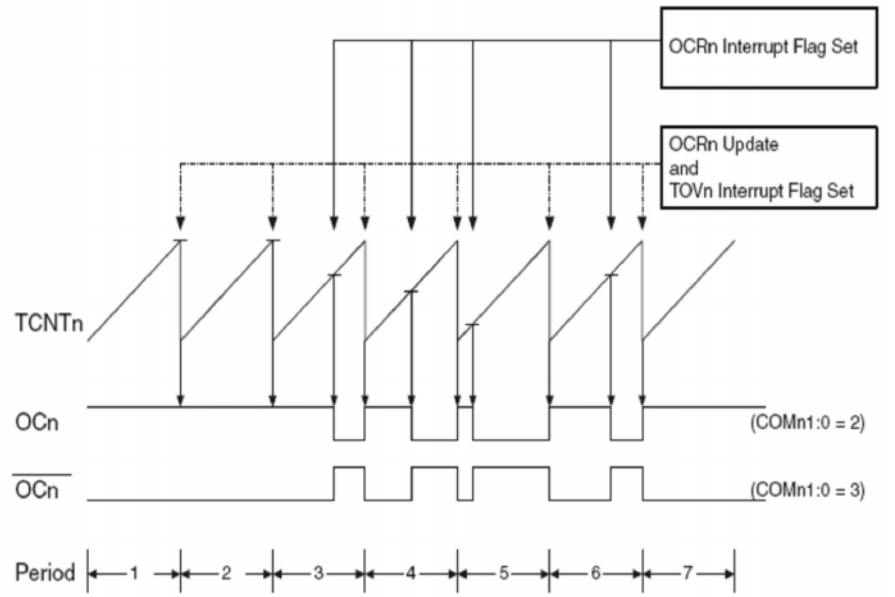
Также как и для нормального режима работы, флаг TOV2 устанавливается на том же такте таймера, когда его значение изменяется с 0xFF на 0x00.

Режим быстрой ШИМ

Режим быстрой широтно-импульсной модуляции (WGM21, WGM20 = 0b11) позволяет генерировать высокочастотные ШИМ-сигналы. От других режимов ШИМ данный отличается однонаправленностью счета. Счетчик изменяет свое значение в направлении от нижнего предела до максимального значения (0xFF), а затем перезагружается значением нижнего предела и счет продолжается снова до максимального значения. Если задан неинвертирующий режим выхода, то при совпадении TCNT2 и OCR2 сигнал OC2 сбрасывается, а при достижении нижнего предела счета устанавливается. Если задан инвертирующий режим, то выход OC2 устанавливается при совпадении и сбрасывается на нижнем пределе счета. За счет однонаправленности счета, рабочая частота для данного режима в два раза выше по сравнению с режимом ШИМ с фазовой коррекцией, где используется двунаправленный счет. Возможность генерации высокочастотных ШИМ сигналов делает использование данного режима полезным в задачах стабилизации питания, выпрямления и цифро-аналогового преобразования. Высокая частота, при этом, позволяет использовать внешние элементы физически малых размеров (индуктивности, конденсаторы), тем самым снижая общую стоимость системы.

В режиме быстрой ШИМ содержимое счетчика инкрементируется пока не достигнет максимального значения (0xFF). Следующим тактовым импульсом счетчик сброситься. Временная диаграмма для режима быстрой ШИМ показана на рис. 50. Значение TCNT2 на временной диаграмме показано в виде графика функции для иллюстрации однонаправленности счета. На диаграмме показаны как инвертированный, так и неинвертированный ШИМ-выходы. Короткой горизонтальной линией показаны точки на графике TCNT2, где совпадают значения OCR2 и TCNT2.

Рисунок 50. Временная диаграмма режима быстрой ШИМ



Флаг переполнения таймер-счетчика TOV0 устанавливается каждый раз, когда счетчик достигает своего максимального значения (0xFF). Если прерывание разрешено, то процедура обработки прерывания может использоваться для обновления сравниваемого значения.

В режиме быстрой ШИМ блок сравнения позволяет генерировать ШИМ-сигналы на выводе OC2. Если COM21:0=0b10, то выходной ШИМ-сигнал не инвертируется, а если COM21:0=0b11, то инвертируется (см. таблицу 44). Фактическое значение OC2 будет присутствовать на выводе порта, если для данного вывода задано выходное направление. ШИМ-сигнал генерируется путем установки (или сброса) сигнала OC2 при совпадении значений OCR2 и TCNT2 и сброса (или установки) сигнала OC2, если счетчик сбрасывается (состояние счетчика изменяется с максимального значения на нижний предел счета).

Частота выходного ШИМ-сигнала может быть вычислена по следующему выражению:

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

где N - переменная, которая задает значение коэффициента предделения (1, 8, 32, 64, 128, 256 или 1024).

Задание предельных значений регистра OCR2 связано с особыми случаями в генерации ШИМ- сигнала в режиме быстрой ШИМ. Если в регистр OCR2 записать значение равное нижнему пределу счета, то через каждые 256(MAX+1) тактов синхронизации таймера на выходе будет генерироваться импульс короткой длительности. Если установить значение OCR2 равным максимальному значению, то выход будет иметь постоянный логический уровень 1 или 0 (зависит от заданного режима выхода битами COM21:0).

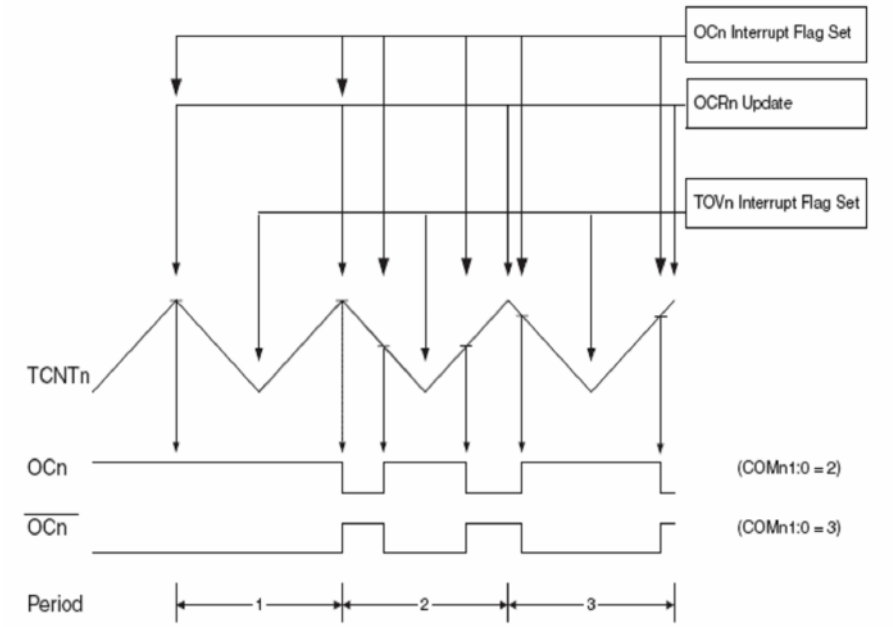
Генерация меандра (50%-ое заполнение импульсов) в данном режиме возможна, если задать режим переключения (инвертирование состояния) выхода OC2 при каждом совпадении (COM21, COM20 = 0b01). Если в регистр OCR2 записать ноль, то будет генерироваться максимальная частота foc2 = fclk_I/O/2. Данная функция похожа на переключение OC2 в режиме CTC за исключением того, что в режиме быстрой ШИМ работает двойная буферизация в блоке сравнения.

Режим широтно-импульсной модуляции с фазовой коррекцией (ШИМ ФК)

Режим ШИМ ФК (WGM21, WGM20 = 0b01) позволяет выполнять фазовую коррекцию ШИМ- сигнала с высокой разрешающей способностью. Режим ШИМ ФК основан на двунаправленной работе таймера-счетчика. Счетчик циклически выполняет счет в направлении от нижнего предела до максимального значения, а затем обратно от максимального значения к нижнему пределу. Если задан неинвертирующий режим выхода компаратора, то выход OC2 сбрасывается/устанавливается при совпадении значений TCNT2 и OCR2 во время прямого/обратного счета. Если задан инвертирующий режим выхода, то, наоборот, во время прямого счета происходит установка, а во время обратного - сброс выхода OC2. При двунаправленной работе максимальная частота ШИМ-сигнала меньше, чем при однонаправленной работе, однако, за счет такой особенности, как симметричность в режимах ШИМ с двунаправленной работой, данные режимы предпочитают использовать при решении задач управления приводами.

Разрешающая способность ШИМ для данного режима фиксирована и равна 8 бит. В режиме ШИМ ФК счетчик инкрементирует свое состояние пока не достигнет максимального значения (0xFF). По достижении максимального значения счетчик реверсируется. Значение TCNT2 остается равным максимальному значению в течение одного такта синхронизации таймера. На рисунке 51 показана временная диаграмма работы таймера-счетчика для режима ШИМ ФК. Значение TCNT2 представлено в виде графика для иллюстрации двунаправленности работы счетчика. На рисунке представлены как неинвертированный, так и инвертированный ШИМ-выходы. Короткие горизонтальные линии указывают точки на графике изменения TCNT2, где совпадают значения OCR2 и TCNT2.

Рисунок 51. Временная диаграмма режима ШИМ ФК



Флаг переполнения таймера-счетчика (TOV2) устанавливается при достижении счетчиком нижнего предела и может использоваться для генерации прерывания.

В режиме ШИМ ФК блок сравнения позволяет генерировать ШИМ-сигналы на выводе OC2. Установка бит COM21,COM20 = 0b10 соответствует неинвертированному режиму генерации ШИМ- сигнала, а COM21, COM20 = 0b11 - инвертированному (см. табл. 45 на странице 118). Фактическое значение OC2 будет присутствовать на соответствующем выводе порта, если для него задано выходное направление. ШИМ-сигнал генерируется путем сброса (установки) сигнала OC2 при совпадении OCR2 и TCNT2 во время прямого счета и путем установки (сброса) сигнала OC2 при совпадении OCR2 и TCNT2 во время обратного счета. Результирующая частота ШИМ-сигнала в режиме ШИМ ФК может быть вычислена по следующему выражению:

$$f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

где N - коэффициент деления предделителя (1, 8, 32, 64, 128, 256 или 1024).

Задание предельных значений регистра OCR2 связано с особыми случаями генерации ШИМ- сигнала в режиме ШИМ ФК. Если значение OCR2 установить равным нижнему пределу, то выход будет постоянно иметь низкий уровень, а если установить равным максимальному значению (0xFF), то выход будет постоянно находиться в высоком состоянии, если задан неинвертирующий режим. Инвертирующему режиму соответствуют противоположные указанным логические уровни.

В самом начале периода 2 на рисунке 51 сигнал OCn переходит из 1 в 0 даже при том, что нет совпадения. Точка этого перехода гарантирует симметричность относительно нижней границы счета. Имеется два случая, в которых изменение логического состояния выхода не зависит от возникновения совпадения:
•OCR2A изменяет свое значение с максимального (0xFF) на другое (неравное 0xFF) (см. рис. 51). Если значение OCR2A равно максимальному, то вывод OCn останется на прежнем уровне как результат совпадения во время обратного счета. Для гарантирования симметричности относительного нижнего предела (0x00) значение OCn при максимальном значении счетчика должно соответствовать результату совпадении при прямом счете.

•Таймер начинает счет с большего значения по сравнению с OCR2A и по этой причине не возникает совпадение и, следовательно, изменение OCn, которое должно было возникнуть во время прямого счета.

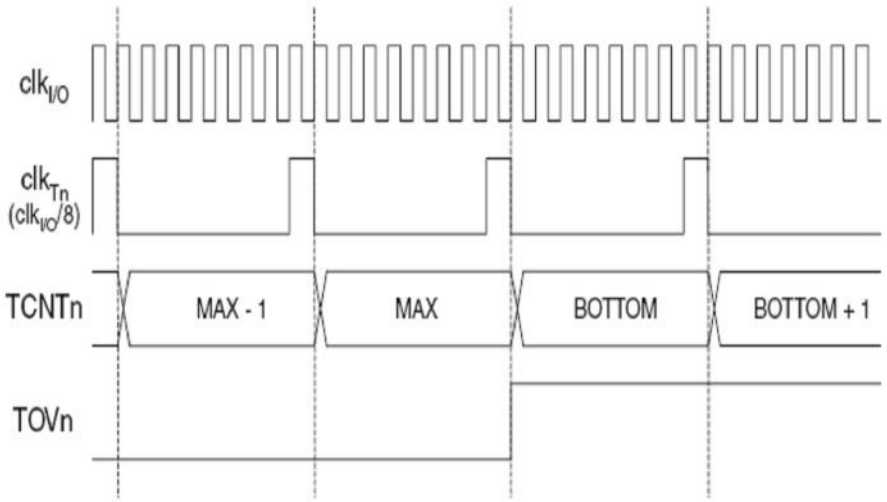
Временные диаграммы таймера-счетчика 2

На рисунке 52 представлена временная диаграмма работы таймер-счетчика 2 без предделения, при этом, тактовый сигнал таймера (clkT2) показан как сигнал разрешения синхронизации. Счетная последовательность показана в области максимального значения счетчика (0xFF).

Рисунок 52. Временная диаграмма таймера-счетчика без предделения

На рисунке 53 показаны аналогичные временные диаграммы, но с разрешенным предделением тактового сигнала.

Рисунок 53. Временная диаграмма таймера-счетчика с предделением на 8 (fclk I/O/8)



На рисунке 54 иллюстрируются моменты установки флагов прерываний, в т.ч. флаг OCF2 для всех режимов, кроме CTC.

Рисунок 54. Временная диаграмма таймера-счетчика с установкой флага OCF2 и предделением на 8 (fclk_I/O/8)

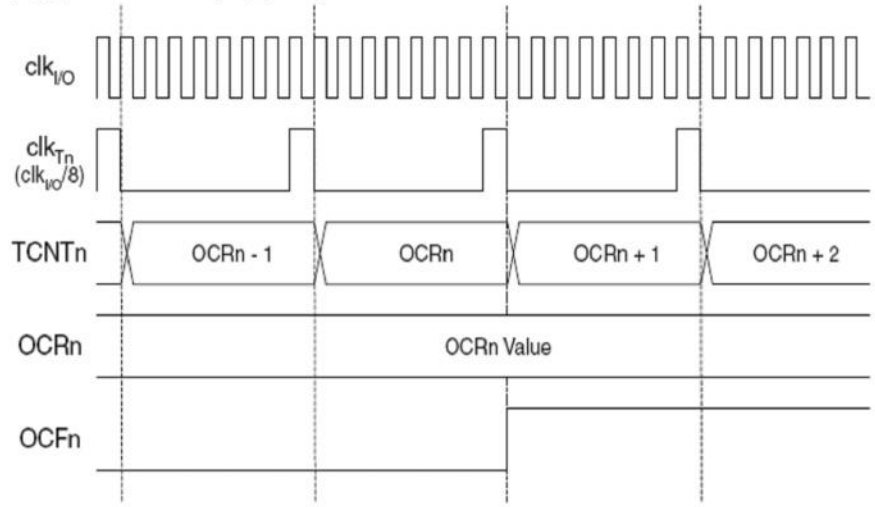
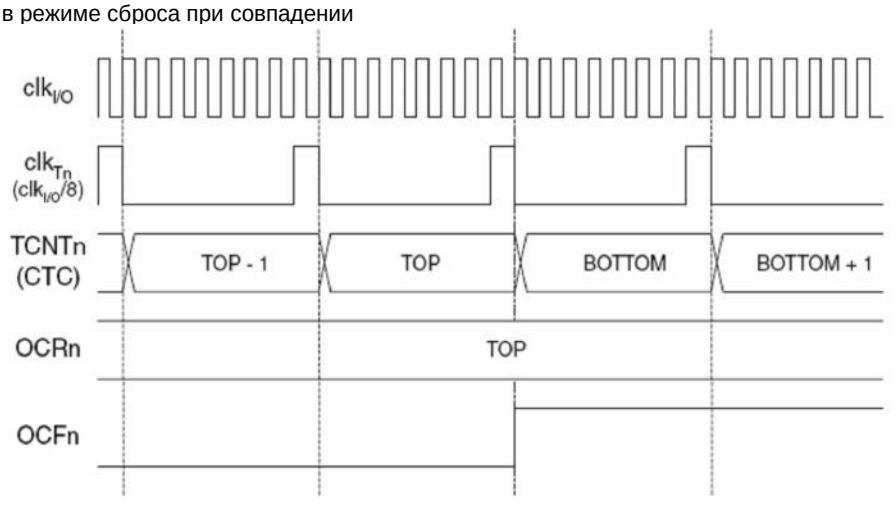


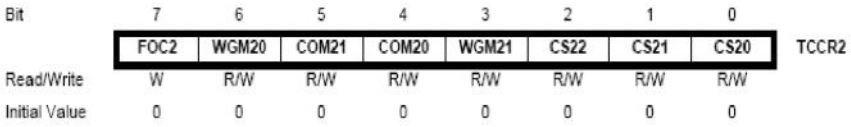
Рисунок 55 показывает установку OCF2 и очистку от TCNT2 в режиме CTC.

Рисунок 55. Временная диаграмма таймер-счетчика с предделением на 8 (fclk_I/O/8) в режиме сброса при совпадении



Описание регистров 8-разрядного таймера-счетчика 2

Регистр управления таймером-счетчиком 2 – TCCR2



Bit	7	6	5	4	3	2	1	0	
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• **Bit 7 – FOC2:** Принудительная установка результата сравнения
Функция бита FOC2 активна только, если с помощью бит WGM задан один из режимов, где нет широтно-импульсной модуляции. Однако в целях совместимости с последующими микроконтроллерами рекомендуется во время записи в регистр TCCR2 в позиции данного бита указывать лог. 0, если таймер работает в одном из режимов с широтно-импульсной модуляцией. Если записать лог. 1 в бит FOC2, то это приводит к принудительной установке результата сравнения на входе блока формирования выходного сигнала. Выход OC2 изменяется в соответствии с установками бит COM21, COM20. Следовательно, значение записанное в COM21, COM20 определяет эффект действия принудительной установки результата сравнения. Обратите также внимание, что бит FOC2 является стробирующим.

Строб FOC2 не генерирует каких-либо прерываний, а также не вызывает сброс таймера в режиме CTC, где регистр OCR2 задает верхний предел счета.
Бит FOC2 всегда считывается как 0.

• **Bit 6,3 – WGM21:0:** WGM01:0: Режим работы таймера-счетчика 2
Данные биты определяют алгоритм счета счетчика, источник, который задает верхний предел счета и тип генерируемых прямоугольных импульсов. Данным таймером поддерживаются следующие режимы работы: нормальный режим, режим сброса при совпадении и два режима с широтно-импульсной модуляцией. В таблице 42 представлены режимы работы таймера-счетчика 2 (см. также "Режимы работы таймера-счетчика 2" на странице 110).

Таблица 42. Описание бит, задающих режим работы таймера-счетчика 2

Номер режима	WGM21 (CTC2)	WGM20 (PWM2)	Наименование режима работы таймера-счетчика 2	Верхний предел счета	Условие обновления содержимого регистра OCR2	Условие установки флага TOV2
0	0	0	Нормальный	0xFF	Сразу после записи в регистр	Достижение максимального значения (0xFF)
1	0	1	ШИМ с фазовой коррекцией	0xFF	Достижение верхнего предела счета	Достижение минимального значения (0x00)
2	1	0	Сброс при совпадении (CTC)	OCR2	Сразу после записи в регистр	Достижение максимального значения (0xFF)
3	1	1	Быстрая ШИМ	0xFF	Достижение минимального значения (0x00)	Достижение максимального значения (0xFF)

Прим.: 1. определение CTC2 и имена разрядов PWM2 являются теперь устаревшими. Используйте определения WGM21:0. Однако, функциональность и расположение этих битов являются совместимыми с предыдущими версиями таймера.

• **Bit 5:4 – COM21:0:** Режим формирования выходного сигнала
Данные биты определяют алгоритм изменения сигнала на выводе OC2. Если значение данных бит ненулевое, то функция вывода OC2 как обычного порта ввода-вывода заменяется на альтернативную. Однако, следует учитывать, что направление этого вывода также управляется через регистр направления данных порта B (DDRB). Поэтому, для разрешения альтернативной функции вывода OC2 также необходимо установить бит 3 (OC2) в регистре DDRB для установки выходного направления. После активизации альтернативной функции назначение бит COM21, COM20 зависит от выбранного режима работы таймера битами WGM21, WGM20. В таблице 43 приведено назначение бит COM21, COM20, если с помощью WGM21, WGM20 задан нормальный режим или режим сброса при совпадении (т.е. режимы без ШИМ).

Таблица 43. Режимы формирования выходного сигнала в режимах работы таймера 2 без ШИМ

COM21	COM20	Описание
0	0	Функция обычного порта ввода-вывода. OC2 отключен.
0	1	Переключение (инвертирование) OC2 при каждом совпадении
1	0	Сброс OC2 при каждом совпадении
1	1	Установка OC2 при каждом совпадении

В таблице 44 приведено назначение бит COM21, COM20 для режима работы таймера-счетчика 2 с быстрой ШИМ (WGM21:0).

Таблица 44. Режимы формирования выходного сигнала в режиме таймера 0 с быстрой ШИМ(1)

COM21	COM20	Описание
0	0	Функция обычного порта ввода-вывода. OC2 отключен.
0	1	Зарезервировано
1	0	Сброс OC2 при совпадении, установка по достижении нижнего предела (неинвертирование)
1	1	Установка OC2 при совпадении, сброс по достижении нижнего предела (инвертирование)

Прим. 1: Имеется особый случай, когда OCR2 = 0xFF и COM21=1. В этом случае возникновение совпадения игнорируется, но сброс или установка по достижении верхнего предела выполняется. См. "Режим быстрой ШИМ" странице 112.

В таблице 45 приведено действие бит COM21, COM20 для режима ШИМ с фазовой коррекцией, заданного с помощью бит WGM21, WGM20.

Таблица 45. Режимы формирования выходного сигнала в режиме ШИМ с фазовой коррекцией(1)

COM21	COM20	Описание
0	0	Функция обычного порта ввода-вывода. OC2 отключен.
0	1	Зарезервировано
1	0	Сброс OC2 при совпадении во время прямого счета. Установка OC2 при совпадении во время обратного счета.
1	1	Установка OC2 при совпадении во время прямого счета. Сброс OC2 при совпадении во время обратного счета.

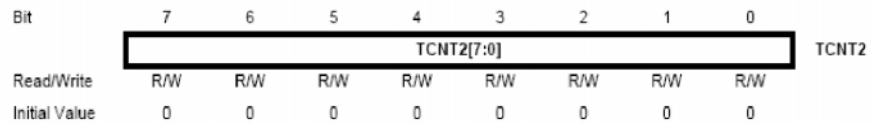
Прим. 1: Существует особый случай, когда OCR2=0xFF и COM21=1. В этом случае сброс при совпадении игнорируется OC2 устанавливается или сбрасывается при достижении верхнего предела (см. также "Режим ШИМ с фазовой коррекцией") на странице 113.

• **Bit 2:0 – CS22:0:** Настройка частоты синхронизации таймера
С помощью трех настроечных бит имеется возможность выбрать различные тактовые частоты, кратные исходной частоте синхронизации (см. табл. 46).

Таблица 46. Выбор частоты синхронизации таймера 2

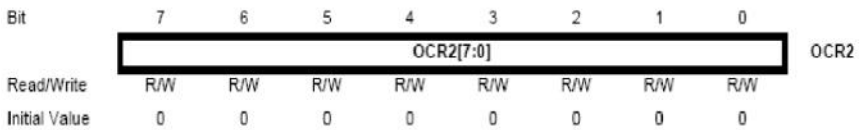
CS22	CS21	CS20	Описание
0	0	0	Нет синхронизации. Таймер-счетчик 0 оставлен.
0	0	1	clkT2S/1 (без предделения)
0	1	0	clkT2S/8 (с предделением)
0	1	1	clkT2S/32 (с предделением)
1	0	0	clkT2S/64 (с предделением)
1	0	1	clkT2S/128 (с предделением)
1	1	0	clkT2S/256 (с предделением)
1	1	1	clkT2S/1024 (с предделением)

Регистр таймера-счетчика – TCNT2



Регистр таймера-счетчика характеризуется двунаправленностью доступа к 8-разрядному счетчику таймера 2. Запись в регистр TCNT2 блокирует отработку возникающего совпадения на следующем после записи такте синхронизации таймера. Изменение содержимого счетчика (TCNT2) во время счета связано с риском потери результата сравнения между TCNT2 и регистром OCR2.

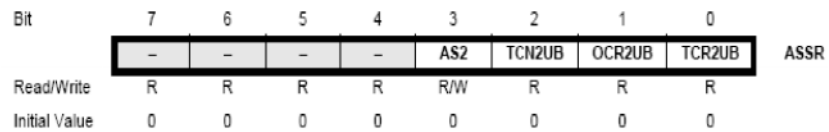
Регистр порога сравнения – OCR2



Регистр порога сравнения содержит 8-разр. значение, которое непрерывно сравнивается цифровым компаратором со значением 8-разр. счетчика (TCNT2). Факт совпадения значений может использоваться для генерации прерывания по выполнению условия сравнения или для генерации прямоугольных импульсов на выводе OC2.

Асинхронная работа таймера-счетчика 2

Регистр асинхронного состояния – ASSR



• **Bit 3 – AS2:** Разрешение асинхронного тактирования таймера-счетчика 2
Если AS2 = 0, то таймер счетчик 2 тактируется сигналом синхронизации ввода-вывода - clkI/O. Если AS2 = 1, то таймер-счетчик 2 тактируется низкочастотным кварцевым генератором, связанного с задающим кварцем через выводы TOSC1 и TOSC2. При изменении значения AS2 содержимое регистров TCNT2, OCR2 и TCCR2 может быть нарушено.

• **Bit 2 – TCN2UB:** Флаг занятости таймера-счетчика 2 при обновлении
Если таймер-счетчик 2 работает асинхронно и выполнена запись в TCNT2, то данный флаг устанавливается. После того, как содержимое TCNT2 обновляется из временного регистра, данный флаг сбрасывается аппаратно. Следовательно, когда TCN2UB=0, в TCNT2 может быть выполнена запись нового значения.

• **Bit 1 – OCR2UB:** Флаг занятости регистра порога сравнения при обновлении
Если таймер-счетчик 2 работает асинхронно и выполнена запись в регистр OCR2, то данный флаг устанавливается. По завершении обновления OCR2 из временного регистра данный флаг сбрасывается аппаратно. Если OCR2UB=0, то это означает готовность регистра OCR2 к записи нового значения.

• **Bit 0 – TCR2UB:** Флаг занятости регистра управления таймера-счетчика 2 при обновлении
Если таймер-счетчик 2 работает асинхронно и выполнена запись в регистр TCCR2, то данный флаг устанавливается. По завершении обновления TCCR2 из временного регистра данный флаг сбрасывается аппаратно. Если TCR2UB=0, то это означает готовность регистра TCCR2 к записи нового значения.

Если выполнить запись в любой из трех регистров таймера-счетчика 2, когда соответствующий флаг занятости установлен, то обновляемое значение может быть нарушено и может стать причиной несанкционированного возникновения прерывания.

Механизм чтения TCNT2, OCR2 и TCCR2 различный. Если выполняется чтение TCNT2, то считывается фактическое значение таймера. Если же считывается значение OCR2 или TCCR2, то фактически считывается содержимое временного регистра.

Асинхронная работа таймера-счетчика 2

Если таймер-счетчик 2 работает асинхронно, то необходимо учесть некоторые особенности. Предупреждение: При переключении между асинхронным и синхронным тактовыми источниками таймера-счетчика содержимое регистров TCNT2, OCR2 и TCCR2 может быть нарушено. Во избежание этого необходимо придерживаться следующей безопасной последовательности переключения:
1. Отключить прерывания таймера-счетчика 2 путем сброса бит OCIE2 и TOIE2.
2. Выбрать необходимый тактовый источник с помощью бита AS2
3. Выполнить запись новых значений в TCNT2, OCR2 и TCCR2.
4. При переходе в асинхронный режим тактирования дождаться сброса флагов TCN2UB, OCR2UB и TCR2UB.
5. Сбросить флаги прерывания таймера-счетчика 2
6. При необходимости разрешить прерывания
• Генератор оптимизирован под использование часового кварцевого резонатора на частоту 32768 Гц. Подключение внешнего тактового сигнала к выводу TOSC1 может сказаться на некорректности работы таймера. Тактовая частота ЦПУ должна быть минимум в четыре раза выше частоты данного генератора. Запись в любой из регистров TCNT2, OCR2 или TCCR2 происходит за два положительных фронта TOSC1, т.к. данные предварительно помещаются во временный регистр, а затем передаются по назначению. Программист должен предусмотреть, чтобы до окончания передачи содержимого временного регистра к назначенному регистру не выполнилась еще одна запись в. этот регистр. Каждый из трех упомянутых регистров имеют свои индивидуальные временные регистры. Это означает, что, например, запись в TCNT2 не влияет на процесс записи в регистр OCR2. Чтобы определить в какой регистр была выполнена запись, реализован регистр асинхронного состояния ASSR.
• Если экономичный режим или расширенный дежурный режим вводится после записи в TCNT2, OCR2 или TCCR2, то программист должен дождаться завершения обновления записанного регистра, в случае если таймер-счетчик 2 используется для пробуждения из этих режимов.

Иначе микроконтроллер перейдет в режим сна прежде чем вступят в силу желаемые изменения. Это особенно важно, если прерывание по результату сравнения таймера-счетчика 2 используется для пробуждения микроконтроллера, т.к. функция отработки условия совпадения блокируется после записи в OCR2 или TCNT2. Если цикл записи не заканчивается и микроконтроллер переводится в режим сна прежде чем OCR2UB станет равным нулю, то микроконтроллер больше не будет прерываться при выполнении условия сравнения и, следовательно, не сможет пробудиться.

• Если таймер-счетчик 2 используется для пробуждения микроконтроллера из экономичного режима или расширенного дежурного режима, то, если требуется перевести данный микроконтроллер снова в один из этих режимов, необходимо учесть несколько особенностей. Для сброса логики прерываний требуется один такт TOSC1. Если интервал времени между пробуждением микроконтроллера и повторным вводом режима сна меньше чем один период TOSC1, то прерывание в дальнейшем не возникнет и микроконтроллер не сможет пробудиться. Если программист не уверен в прохождении достаточного времени перед повторным вводом в экономичный режим или расширенный дежурный режим, то необходимо придерживаться следующей последовательности действий, которая гарантирует прохождение одного периода TOSC1:

1. Запись значения в TCCR2, TCNT2 или OCR2.
2. Ожидание сброса соответствующего флага занятости при обновлении в регистре ASSR.
3. Ввод экономичного или расширенного дежурного режима.

• Если выбрана асинхронная работа, то генератор на 32768 Гц таймера-счетчика 0 находится постоянно включенным, за исключением режима выключения и дежурного режима микроконтроллера. После сброса при подаче питания или пробуждения из режима выключения и дежурного режима программист должен учитывать, что для возобновления нормальной стабильной работы данного генератора требуется минимум 1 секунда. Таким образом, программисту рекомендуется включить задержку минимум на 1 сек. Перед использованием таймера-счетчика 2 после подачи питания или выхода из режима выключения или дежурного режима. Содержимое всех регистров таймера-счетчика 2 необходимо рассматривать как потерянное после подачи питания или пробуждения из указанных выше режимов из-за нестабильности тактового сигнала при запуске независимо от того, какой асинхронный источник используется (генератор или внешний сигнал на выводе TOSC1).

• Выход микроконтроллера из экономичного и расширенного дежурного режимов при асинхронном тактирования таймера-счетчика 2 происходит в следующей последовательности. Если выполняется условие прерывания, то инициируется процесс пробуждения следующим тактом синхронизации таймера, т.е. таймер минимум на 1 изменит свое состояние, прежде чем процессор получит доступ к его состоянию. После пробуждения микроконтроллер задерживается на 4 такта синхронизации ЦПУ, а затем переходит на вектор обработки прерывания, а по завершении процедуры его обработки возвращается к выполнению инструкции, следующей за SLEEP.

• Чтение регистра TCNT2 сразу после пробуждения из экономичного режима (Power-save) может дать некорректный результат. Поскольку TCNT2 тактируется от асинхронного источника TOSC, то чтение TCNT2 должно быть выполнено через регистр, синхронизированный с внутренней синхронизацией ввода-вывода. Синхронизация выполняется каждый нарастающий фронт на TOSC1. При пробуждении из экономичного режима активизируется снова синхронизация ввода-вывода (clkI/O) и при чтении TCNT2 фактически будет считываться предыдущее значение (которое записано перед вводом в режим сна) до следующего нарастающего фронта на TOSC1. Фаза тактового сигнала TOSC после выхода из экономичного режима непредсказуема, т.к. зависит от момента пробуждения микроконтроллера. С учетом этого, при чтении содержимого TCNT2, рекомендуется соблюдать следующую последовательность действий:

1. По усмотрению записать произвольное значение или в регистр OCR2 или в TCCR2.
2. Дождаться сброса флага занятости при обновлении, соответствующего выбранному в п.1 регистру.
3. Выполнить чтение TCNT2.

• Во время асинхронной работы синхронизация флагов прерываний асинхронного таймера требует три такта синхронизации ЦПУ плюс один такт синхронизации таймера. Таким образом, как минимум таймер должен изменить свое состояние на 1 прежде чем процессор сможет считать его содержимое, вызывающее установку флагов прерываний. Выход генератора прямоугольных импульсов OC2 привязан к синхронизации таймера и не синхронизирован с тактированием ЦПУ.

Регистр маски прерываний таймеров-счетчиков – TIMSK

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• **Bit 7 – OCIE2:** Разрешение прерывания по результату сравнения таймера-счетчика 2
Если OCIE2=1, а также установлен бит I в регистре статуса, то прерывание по результату сравнения таймера-счетчика 2 активизируется. В этом случае прерывание возникает, если обнаруживается совпадение значения таймера-счетчика 2 с порогом сравнения, т.е. когда установлен флаг OCF2 в регистре флагов прерываний таймеров- счетчиков TIFR.

• **Bit 6 – TOIE2:** Разрешение прерывания по переполнению таймера-счетчика 2
Если TOIE2=1, а также установлен бит I в регистре статуса, то прерывание по переполнению таймера-счетчика 2 разрешается. В этом случае запрос на прерывание генерируется, если обнаруживается переполнение таймера-счетчика 2, т.е. когда установлен флаг TOV2 в регистре флагов прерываний таймеров-счетчиков TIFR.

Регистр флагов прерываний таймеров-счетчиков – TIFR

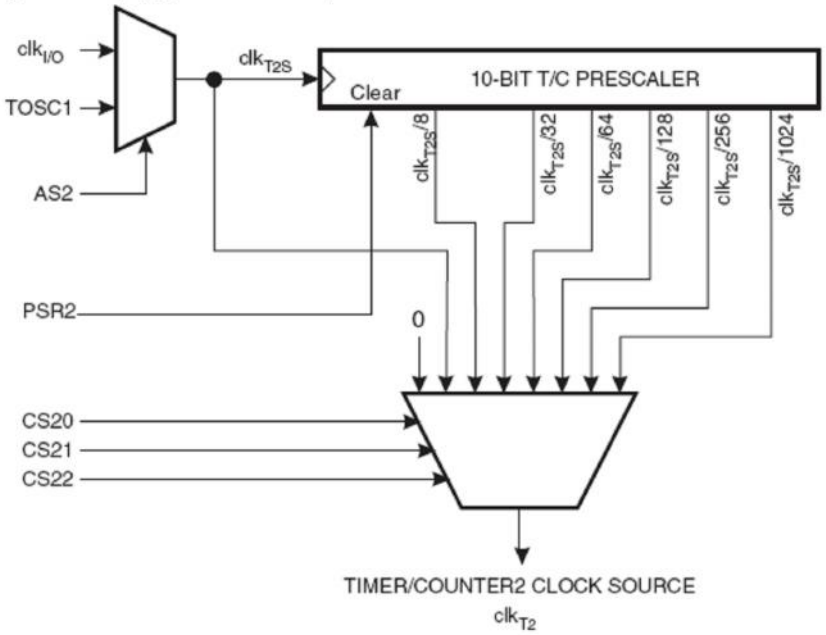
Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• **Bit 7 – OCF2:** Флаг совпадения таймера-счетчика 2
OCF2 равен лог. 1, если обнаруживается совпадением между значением таймера-счетчика 2 и данными в регистре OCR2 (регистр порога сравнения). OCF2 сбрасывается аппаратно при переходе на соответствующий вектор прерывания. Альтернативно, флаг OCF2 может быть сброшен путем записи в него лог. 1. Если установлены бит I в регистре SREG, бит OCIE2 (разрешено прерывание по выполнению условия сравнения таймера- счетчика 2) и флаг OCF2, то генерируется прерывание по выполнению условия сравнения таймера-счетчика 2.

• **Bit 6 – TOV2:** Флаг переполнения таймера-счетчика 2
Флаг TOV2 устанавливается, если в таймере-счетчике 2 возникает переполнение. Флаг TOV2 сбрасывается аппаратно при переходе на соответствующий вектор прерывания. Альтернативно, флаг TOV2 сбрасывается путем записи в него лог. 1. Если установлены бит I в регистре SREG, бит TOIE2 (разрешено прерывание по переполнению таймера- счетчика 2) и флаг TOV2, то генерируется прерывание по переполнению таймера-счетчика 2. В режиме ШИМ данный флаг устанавливается, если таймер-счетчик 2 изменяет направление счета на значении 0x00.

Предделитель таймера-счетчика 2

Рисунок 56. Предделитель таймера-счетчика 2



Тактовый источник таймера-счетчика 0 обозначен как clkT2s. По умолчанию clkT2s подключен к системному источнику синхронизации ввода-вывода clkI/O. Путем установки бита AS2 в регистре ASSR таймер-счетчик 2 тактируется асинхронно с вывода TOSC1. Данная функция позволяет использовать таймер-счетчик 2 в качестве часов реального времени (RTC). Если AS2=1, то выводы TOSC1 и TOSC2 более не выполняют функции линий порта B, а между ними может быть подключен кварцевый резонатор в качестве отдельного тактового источника таймера-счетчика 2. Генератор оптимизирован под использование кварца на частоту 32768 Гц. Подключение к выводу TOSC1 внешнего тактового источника не рекомендуется.

Предделитель таймера-счетчика 2 позволяет выбрать следующие тактовые сигналы: clkT2S/8, clkT2S/32, clkT2S/64, clkT2S/128, clkT2S/256 и clkT2S/1024. Кроме того, имеется возможность остановить синхронизацию. Установка бита PSR2 в регистре SFIOR сбрасывает предделитель. Данная функция позволяет программисту работать с более прогнозируемым поведением предделителя.

Регистр специальных функций ввода-вывода SFIOR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

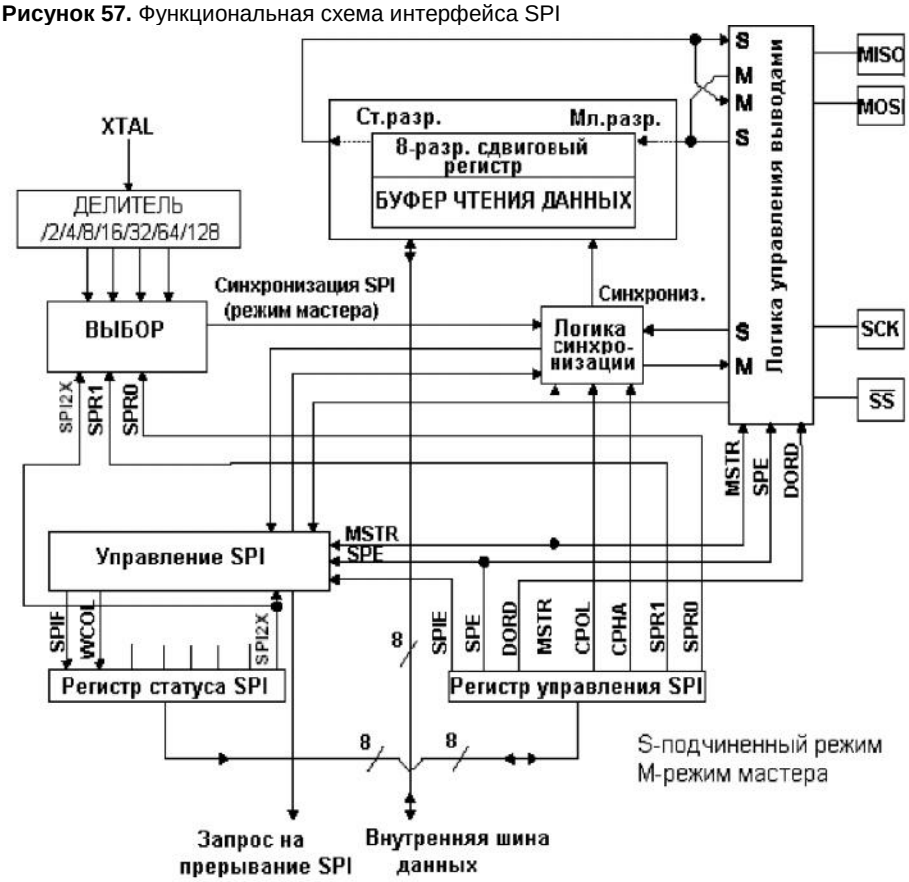
• **Bit 1 – PSR2:** Сброс предделителя таймера-счетчика 2
Если данный бит равен лог. 1, то предделитель таймера-счетчика 2 сбрасывается. Данный бит обычно сбрасывается аппаратно сразу после установки.
Запись нуля в этот бит не будет иметь никакого эффекта. Этот бит будет всегда читаться как нуль, если таймер-счетчик 2 будет синхронизирован внутренней тактовой частотой ЦП. Если данный бит устанавливается, когда таймер-счетчик 2 работает в асинхронном режиме, то он остается равным 1 пока не сбросится предделитель таймера-счетчика 2.

Последовательный периферийный интерфейс – SPI

Интерфейс SPI позволяет организовать последовательную синхронную высокоскоростную передачу данных между ATmega8 и другим периферийным устройством или между несколькими AVR-микроконтроллерами.
SPI ATmega8 включает следующие функции:

- **Полнодуплексная, трехпроводная синхронная передача данных**
- **Ведущая или подчиненная работа**
- **Передача первым младшего(LSB) или старшего(MSB) бита**
- **Семь программируемых скоростей связи**
- **Флаг прерывания для индикации окончания передачи данных**
- **Защитный флаг при повторной записи**
- **Пробуждение из режима холостого хода (Idle)**
- **Режим ведущего (мастера) SPI с удвоением скорости (CK/2)**

Рисунок 57. Функциональная схема интерфейса SPI

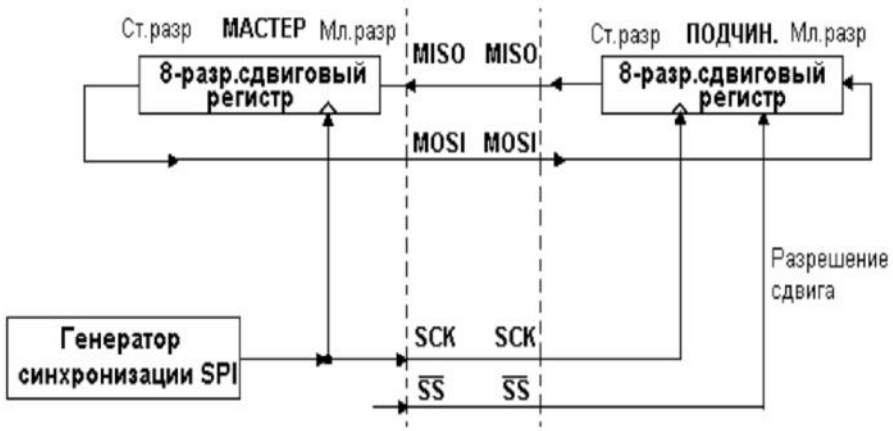


Прим.: Расположение выводов интерфейса SPI представлено на рис. 2 и в таблице 22 на странице 58.

Внешние соединения между ведущим (мастером) и подчиненным ЦПУ через интерфейс SPI показаны на рисунке 58. Система состоит из двух сдвиговых регистров и генератора ведущей синхронизации. Ведущий SPI инициирует сеанс связи подачей низкого уровня на вход SS того подчиненного устройства, с которым необходимо обмениваться данными. Оба респондента (ведущий и подчиненный) подготавливают данные к передаче в своем сдвиговом регистре, при этом на стороне ведущего генерируются также импульсы синхронизации на линии SCK. По линии MOSI всегда осуществляется передача данных от ведущего к подчиненному, а по MISO, наоборот, от подчиненного к мастеру. По окончании передачи каждого пакета данных ведущий SPI должен засинхронизировать подчиненный путем подачи высокого уровня на линию SS (выбор подчиненного интерфейса).

Если SPI настроен как ведущий (мастер), то управление линией SS происходит не автоматически. Данная операция должна быть выполнена программно перед началом сеанса связи. После этого, запись в регистр данных SPI инициирует генерацию синхронизации и аппаратный сдвиг 8-ми разрядов в подчиненное устройство. По окончании сдвига одного байта генератор синхронизации SPI останавливается, при этом устанавливая флаг окончания передачи (SPIF). Если установлен бит SPIE в регистре SPCR, то разрешается прерывание SPI и по окончании передачи байта будет генерирован запрос на прерывание. Мастер может продолжить сдвигать следующий байт, если записать его в регистр SPDR, или подать сигнал окончания пакета путем установки низкого уровня на линии SS. Последний принятый байт сохраняется в буферном регистре. В режиме подчиненного, интерфейс SPI находится в состоянии ожидания, в котором MISO переводится в третье состояние, до тех пор, пока на выводе SS присутствует высокий уровень. В этом состоянии программа может обновлять содержимое регистра данных SPI (SPDR), но при этом входящие импульсы синхронизации не сдвигают данные до подачи низкого уровня на вывод SS. После того как один байт был полностью сдвинут, устанавливается флаг окончания передачи SPIF. Если установлен бит разрешения прерывания SPI (SPIE) в регистре SPCR, то установка флага SPIF приводит к генерации запроса на прерывание. Подчиненный может продолжать размещать новые данные для передачи в регистр SPDR перед чтением входящих данных. Последний принятый байт хранится в буферном регистре.

Рисунок 58. Внешнее соединение ведущего (мастера) и подчиненного SPI



В направлении передачи данных система выполнена как однобуферная, а в направлении приема используется двойная буферизация. Это означает, что передаваемые байты не могут быть записаны в регистр данных SPI, прежде чем полностью завершится цикл сдвига. Во время приема данных необходимо следить, чтобы принятая посылка была считана из регистра данных SPI, прежде чем завершится цикл входящего сдвига новой посылки. В противном случае первый байт будет потерян.

В подчиненном режиме SPI управляющая логика осуществляет выборку входящего сигнала SCK. Чтобы гарантировать корректность выборки тактового сигнала минимальные низкие и высокие периоды должны быть:
Низкий период: дольше чем 2 цикла тактовой частоты ЦП
Высокий период: дольше чем 2 цикла тактовой частоты ЦП.
Если работа SPI разрешена, то разрешается альтернативное направление выводов MOSI, MISO, SCK и SS (см. табл. 47).

Таблица 47. Направление выводов SPI(1)

Вывод	Направление для ведущего SPI	Направление для подчиненного SPI
MOSI	Определяется пользователем	Вход
MISO	Вход	Определяется пользователем
SCK	Определяется пользователем	Вход
SS	Определяется пользователем	Вход

Прим.1: См. "Альтернативные функции порта B", где подробно описано как установить направление на выводах порта SPI.стр.58

В следующих примерах показаны инициализация SPI как мастера и организация простой передачи данных. В данных примерах DDR_SPI должен быть заменен на имя фактического регистра направления данных, управляющий выводами интерфейса SPI . DD_MOSI, DD_MISO и DD_SCK также должны быть заменены на имена соответствующих бит регистров направления данных, связанных с этими выводами. Например, если MOSI размещен на выв. PB5, то DD_MOSI необходимо заменить на DDB5, а DDR_SPI на DDRB.

Пример кода на Ассемблере (1)

```
SPI_MasterInit:
; Установка MOSI и SCK на вывод, все остальные на ввод
ldi r17,(1<<DD_MOSI)|(1<<DD_SCK)
out DDR_SPI,r17
; Разрешение SPI в режиме мастера, установка скорости связи fck/16
ldi r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
out SPCR,r17
ret
SPI_MasterTransmit:
; Запуск передачи данных (r16)
out SPDR,r16
Wait_Transmit:
; Ожидание завершения передачи данных
sbis SPSR,SPIF
rjmp Wait_Transmit
ret
```

Пример кода на Си(1)

```
void SPI_MasterInit(void)
{
/* Установка MOSI и SCK на вывод, все остальные на ввод */
DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
/* Разрешение SPI в режиме мастера, установка скорости связи fck/16
*/SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}
void SPI_MasterTransmit(char cData)
{
/* Запуск передачи данных */
SPDR = cData;
/* Ожидание завершения передачи данных */
while(!(SPSR & (1<<SPIF)))
;
}
```

Прим.1: В примерах предполагается, что подключен файл специфических заголовков.

В следующем примере показано как инициализировать SPI как подчиненного и как выполнить простой прием данных.

Пример кода на Ассемблере (1)

```
SPI_SlaveInit:
; Установка MISO на вывод и всех ост. на ввод
ldi r17,(1<<DD_MISO)
out DDR_SPI,r17
; Разрешение SPI
ldi r17,(1<<SPE)
out SPCR,r17
ret
SPI_SlaveReceive:
; Ожидание завершения передачи
sbis SPSR,SPIF
rjmp SPI_SlaveReceive
; Чтение принятых данных и выход из процедуры
in r16,SPDR
ret
```

C Code Example(1)

```
void SPI_SlaveInit(void)
{
/* Установка MISO на вывод и всех ост. на ввод */
DDR_SPI = (1<<DD_MISO);
/* Разрешение SPI */
SPCR = (1<<SPE);
}
char SPI_SlaveReceive(void)
{
/* Ожидание завершения передачи */
while(!(SPSR & (1<<SPIF)));
/* Чтение принятых данных и выход из процедуры */
return SPDR;
}
```

Прим.1: В примерах предполагается, что подключен файл специфических заголовков.

Функционирование вывода SS

Подчиненный режим

После перевода SPI в режим подчиненного вывод SS всегда работает как вход. В этом случае SPI активизируется, если на вход SS подать низкий уровень, а вывод MISO становится выходом, если так установит пользователь. Все остальные выводы работают как входы. Если на вход SS подать высокий уровень, то все выводы станут входами и SPI перейдет в пассивное состояние, в котором блокируется прием входящих данных. Обратите внимание, что логика SPI сбрасывается как только на вывод SS подается высокий лог. уровень.

Вывод SS удобно использовать для пакетной/байтной синхронизации, что позволяет поддержать синхронность работы подчиненного счетчика бит и ведущего генератора синхронизации. Если на вывод SS подать высокий лог. уровень, то подчиненный SPI сбросит передающую и приемную логику и потеряет любые не полностью принятые данные в сдвиговом регистре.

Ведущий режим

Если SPI настроен как мастер (установлен бит MSTR в SPCR), то пользователь может задать желаемое направление вывода SS.
Если SS настроен на вывод, то он работает как обычная линия цифрового вывода и не оказывает влияния на систему SPI. Обычно он используется для управления выводом SS подчиненного SPI.

Если SS настроить как вход, то на нем должен присутствовать высокий лог. уровень, чтобы гарантировать работу ведущего SPI. Если SPI настроен как мастер, у которого выв. SS настроен как вход, то подача на этот вход низкого уровня внешней схемой будет интерпретирована как перевод в подчиненный режим по запросу другого ведущего SPI, после чего начнется передача данных. Для того чтобы избежать конфликтной ситуации система SPI выполняет следующие действия:

1. SPI переводится в подчиненный режим сбросом бита MSTR в регистре SPCR. В результате SPI становится подчиненным, а MOSI и SCK конфигурируются как входы.
2. Устанавливается SPIF в SPSR и, если разрешено прерывание SPI и установлен бит I в регистре SREG, то выполняется процедура обработки прерывания.

Таким образом, если используется передача SPI в режиме мастера с управлением по прерываниям и предусмотрена возможность подачи низкого уровня на вход SS, то при генерации прерывания необходимо всегда проверять состояние бита MSTR. Если MSTR оказался сброшенным, то это означает, что SPI был переведен в подчиненный режим внешним устройством и пользователь должен предусмотреть возобновление ведущего режима SPI программным путем.

Регистр управления SPI – SPCR

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• **Bit 7 – SPIE:** Разрешение прерывания SPI
Если установлен флаг SPIF в регистре SPSR и установлен бит общего разрешения прерываний I в регистре SREG, то установка данного бита приведет к исполнению процедуры обработки прерывания по SP

• **Bit 6 – SPE:** Разрешение SPI
Если в SPE записать лог. 1, то разрешается работа SPI. Данный бит должен быть установлен, если необходимо использовать SPI независимо от того в каком режиме он будет работать.

• **Bit 5 – DORD:** Порядок сдвига данных
Если DORD=1, то при передаче слова данных первым передается младший разряд. Если же DORD=0, то первым передается старший разряд.

• **Bit 4 – MSTR:** Выбор ведущего/подчиненного
Если в данный бит записана лог. 1, то SPI работает как ведущий (мастер), иначе (MSTR=0) как подчиненный.

Если SS настроен как вход и к нему приложен низкий уровень, когда MSTR был равен 1, то бит MSTR автоматически сбрасывается и устанавливается флаг прерывания SPIF в регистре SPSR. Для возобновления ведущего режима SPI пользователь должен предусмотреть программную установку бита MSTR.

• **Bit 3 – CPOL:** Полярность синхронизации
Если данный бит равен лог. 1, то SCK имеет высокий уровень в состоянии ожидания. Если CPOL=0, то SCK имеет низкий уровень в состоянии ожидания. См. примеры, иллюстрирующие отличия в полярности синхронизации, на рис. 59 и 60. Ниже обобщено функционирование CPOL:

Таблица 48. Результат действия CPOL

CPOL	Передний фронт	Задний фронт
0	Нарастающий	Спадающий
1	Спадающий	Нарастающий

• **Bit 2 – CPHA:** Фаза синхронизации
Значение бита фазы синхронизации (CPHA) определяет по какому фронту SCK происходит выборка данных: по переднему или заднему. Примеры действия различных установок CPHA приведены на рисунках 59 и 60. Действие CPHA подытожено ниже:
Таблица 49. Результат действия бита CPHA

CPHA	Передний фронт	Задний фронт
0	Выборка	Установка
1	Установка	Выборка

• **Bits 1, 0 – SPR1, SPR0:** Биты 1 и 0 выбора частоты синхронизации SPI
Данные биты задают частоту синхронизации на выводе SCK в режиме мастера. SPR1 и SPR0 не оказывают никакого влияния в режиме подчиненного. Связь между частой SCK и частотой генератора синхронизации fosc показана ниже в таблице:
Таблица 50. Связь между частотами SCK и генератора

SPI2X	SPR1	SPR0	Частота SCK
0	0	0	fosc /4
0	0	1	fosc /16
0	1	0	fosc /64
0	1	1	fosc /128
1	0	0	fosc /2
1	0	1	fosc /8
1	1	0	fosc /32
1	1	1	fosc /64

Регистр статуса SPI – SPSR

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• **Bit 7 – SPIF:** Флаг прерывания по SPI
Флаг SPIF устанавливается по завершении последовательной передачи. Прерывание генерируется в том случае, если установлен бит SPIE в регистре SPCR и разрешены общие прерывания. Если SS настроен как вход и к нему приложен низкий уровень, то, если SPI находился в режиме мастера, также установится флаг SPIF. SPIF сбрасывается аппаратно при переходе на соответствующий вектор прерывания. Альтернативно, бит SPIF сбрасывается при первом чтении регистра статуса SPI с установленным флагом SPIF, а также во время доступа к регистру данных SPI (SPDR).

• **Bit 6 – WCOL:** Флаг повторной записи
Бит WCOL устанавливается, если выполнена запись в регистр данных SPI (SPDR) во время передачи данных. Бит WCOL (а также бит SPIF) сбрасывается при первом чтении регистра статуса SPI с установленным WCOL, а также во время доступа к регистру данных SPI.

• **Bit 5..1 – Res:** зарезервированные биты
В ATmega8 данные биты не используются и всегда считываются как 0.

• **Bit 0 – SPI2X:** Бит удвоения скорости SPI
Если в данный бит записать лог. 1 то скорость работы SPI (частота SCK) удвоится, если SPI находится в режиме мастера (см. табл. 72). Это означает, что минимальный период SCK будет равен двум периодам синхронизации ЦПУ. Если SPI работает как подчиненный, то работа SPI гарантирована только на частоте fosc /4 или менее.

Интерфейс SPI в ATmega8 также используется для чтения или программирования памяти программ и EEPROM. См. также "Последовательное программирование"страница 237.

Регистр данных SPI – SPDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

Регистр данных SPI имеет доступ на чтение и запись и предназначен для обмена данными между файлом регистров (r0…r31) и сдвиговым регистром SPI. Запись в данный регистр инициирует передачу данных. При чтении данного регистра фактически считывается содержимое приемного буфера сдвигового регистра.

Режимы передачи данных

Комбинация бит CPHA и CPOL задает четыре возможных режима последовательной передачи данных. Форматы передачи данных для SPI представлены в таблице 51, а их временные диаграммы показаны на рис. 59 и 60. Биты данных выводятся сдвигом и фиксируются на входе противоположными фронтами синхросигнала SCK, тем самым гарантируя достаточное время на установление сигналов данных. Таким образом, можно обобщить информацию из табл. 48 и 49 и представить ее в следующем виде:

Таблица 51. Функциональные возможности CPOL и CPHA

	Передний фронт	Задний фронт	Режим SPI
CPOL = 0, CPHA = 0	Выборка нарастающим фронтом	Установка данных падающим фронтом	0
CPOL = 0, CPHA = 1	Установка данных нарастающим фронтом	Выборка падающим фронтом	1
CPOL = 1, CPHA = 0	Выборка падающим фронтом	Установка данных нарастающим фронтом	2
CPOL = 1, CPHA = 1	Установка данных падающим фронтом	Выборка нарастающим фронтом	3

Рисунок 59. Формат передачи данных SPI с CPHA = 0

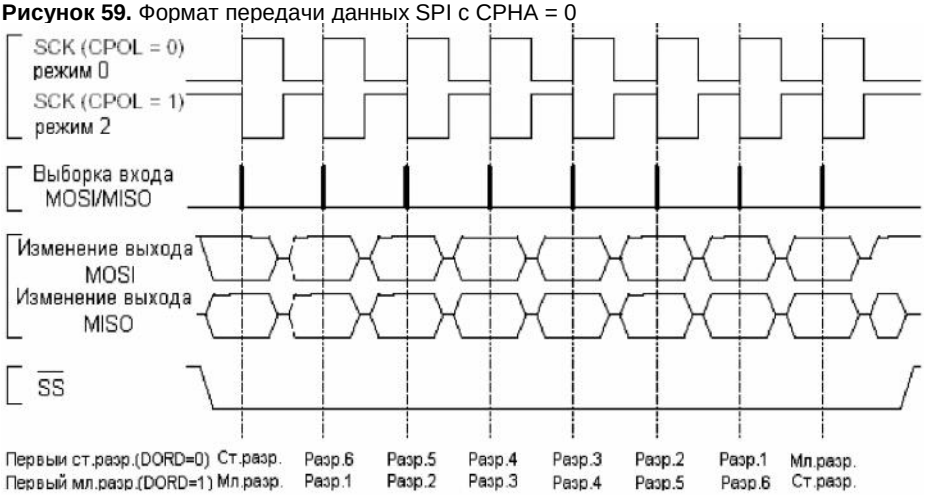
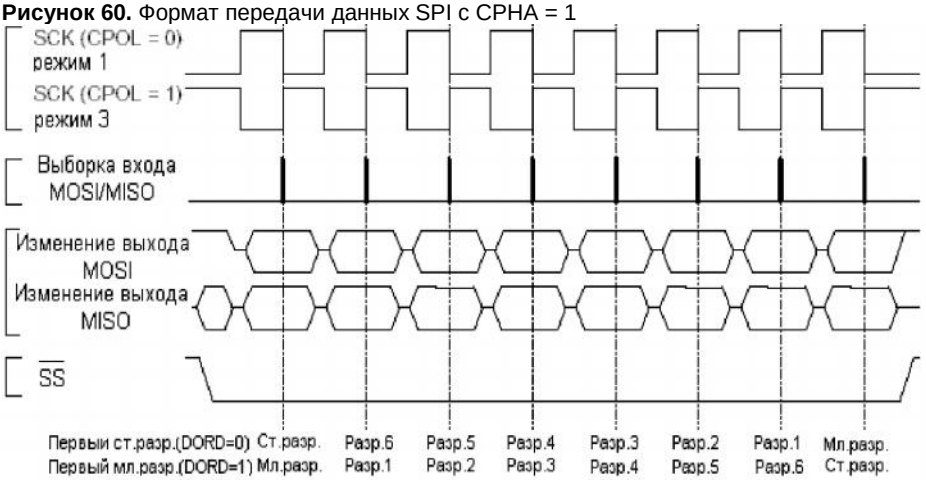


Рисунок 60. Формат передачи данных SPI с CPHA = 1



УСАПП

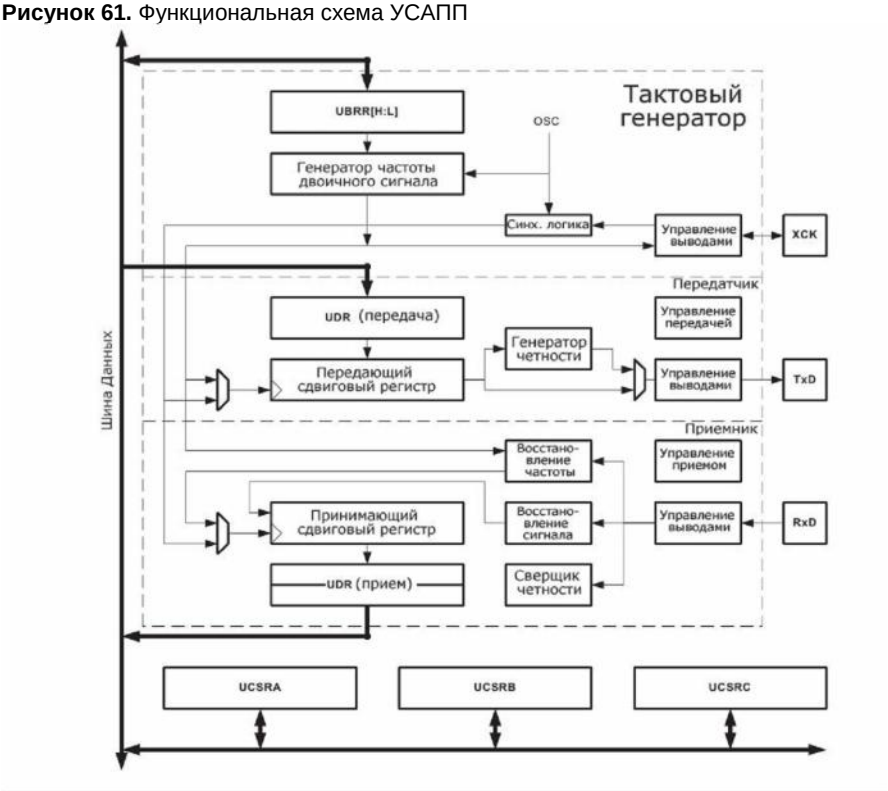
Универсальный синхронный и асинхронный последовательный приемопередатчик (УСАПП) предназначен для организации гибкой последовательной связи.
Отличительные особенности:

- **Полнодуплексная работа (раздельные регистры последовательного приема и передачи)**
- **Асинхронная или синхронная работа**
- **Ведущее или подчиненное тактирование связи в синхронном режиме работы**
- **Высокая разрешающая способность генератора скорости связи**
- **Поддержка формата передаваемых данных с 5, 6, 7, 8 или 9 битами данных и 1 или 2 стоп-битами**
- **Аппаратная генерация и проверка бита паритета (четность/нечетность)**
- **Определение переполнения данных**
- **Определение ошибки в структуре посылки**
- **Фильтрация шума с обнаружением ложного старт-бита и цифровым ФНЧ**
- **Три раздельных прерывания по завершении передачи, освобождении регистра передаваемых данных и завершении приема**
- **Режим многопроцессорной связи**
- **Режим удвоения скорости связи в асинхронном режиме**

Краткий обзор

На рисунке 61 представлена упрощенная функциональная схема УСАПП. На рисунке жирным шрифтом выделены регистры и выводы УСАПП.

Рисунок 61. Функциональная схема УСАПП



Прим. : Расположение выводов УСАПП см. на рисунке 2, табл. 30 на странице 64 и 29 странице 64.

На рисунке 61 пунктирной линией выделены три основных блока УСАПП: тактовый генератор, передатчик и приемник. Регистры управления используются всеми блоками. Логика тактового генератора состоит из логики синхронизации, связанной с внешним тактовым входом (используется в подчиненном режиме) и генератора скорости связи. Вывод XCK (синхронизация передачи) используется только в режиме синхронной передачи. Передатчик состоит из одного буфера записи, последовательного сдвигового регистра, генератора паритета и управляющей логики, которая поддерживает различные форматы последовательной посылки. Буфер записи позволяет непрерывно передавать данные без каких-либо задержек между передачей посылок.
Приемник является более сложным блоком УСАПП, т.к. в его состав входят модули обнаружения данных и синхронизации. Модули обнаружения необходимы для асинхронного приема данных.
Помимо модулей обнаружения в приемник входит устройство проверки паритета, сдвиговый регистр, и двухуровневый приемный буфер (UDR). Приемник поддерживает те же последовательные форматы, что и передатчик, и может определить ошибку в посылке (кадре), переполнение данных и ошибку паритета.

Совместимость УСАПП с УАПП других AVR-микроконтроллеров

УСАПП полностью совместим с УАПП AVR-микроконтроллеров по следующим позициям:

- Расположение бит внутри всех регистров УСАПП
- Генерация скорости связи
- Работа передатчика
- Функционирование буфера передатчика
- Работа приемника

Однако в схеме буферизации приемника реализовано два улучшения, которые в некоторых случаях может повлиять на совместимость:

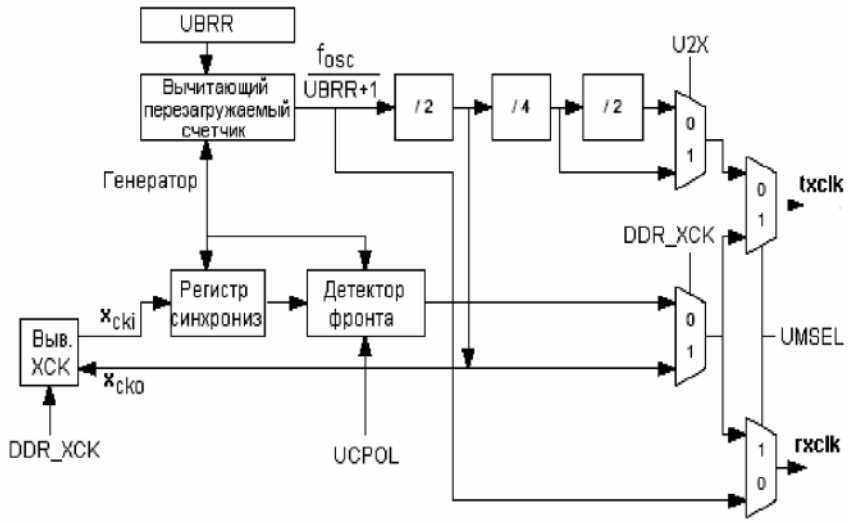
- Добавлен второй буферный регистр. Два буферных регистра работают как циклический буфер FIFO. Поэтому, UDR необходимо опрашивать только один раз при каждом получении данных! Более важным является тот факт, что флаги ошибок (FE и DOR), а также 9-ый бит данных (RXB8) также буферизованы вместе с данными в приемном буфере. Поэтому, состояние статусных бит необходимо всегда считывать перед чтением регистра UDR. В противном случае состояние флагов ошибок будет потеряно, т.к. будет изменено состояние буфера.
- Сдвиговый регистр приемника действует как трехуровневый буфер. Этим обеспечивается возможность сохранения принятых данных в последовательном сдвиговом регистре (см. рисунок 61) до определения нового старт-бита, если буферные регистры заполнены. Таким образом, УСАПП характеризуется более высокой стойкостью к выполнению условия
ошибки по переполнению данных (DOR).

У следующих управляющих битах изменены наименования, но сохранены назначение, механизм действия и расположение в регистре:
CHR9 заменен на UCSZ2
OR заменен на DOR

Генерация тактовых импульсов

Логика генерации тактовых импульсов формирует основную синхронизацию приемника и передатчика. УСАПП поддерживает четыре режима работы синхронизации: нормальная асинхронная, асинхронная с удвоением скорости, ведущая синхронная и подчиненная синхронная. Бит UMSEL в регистре С управления и статуса (UCSRC) позволяют выбрать асинхронную или синхронную работу. Удвоение скорости (только в асинхронном режиме) управляется битом U2X в регистре UCSRA. При использовании синхронного режима (UMSEL = 1) соответствующий бит в регистре направления данных для вывода XCK (DDR_XCK) задает будет ли синхронизация внутренней (ведущий режим) или внешней (подчиненный режим). Вывод XCK активен только при использовании синхронного режима.
На рисунке 62 показана функциональная схема логики синхронизации.

Рисунок 62. Функциональная схема логики синхронизации УСАПП



Описание сигналов:
txclk - синхронизация передатчика (внутренний сигнал)
rxclk - основная синхронизация приемника (внутренний сигнал)
xcki - вход от вывода XCK (внутренний сигнал). Используется для синхронной подчиненной
работы.
xcko - выход синхронизации к выводу XCK (внутренний сигнал). Используется в ведущем
синхронном режиме.
fosc - вывод частоты XTAL (системная синхронизация).

Генерация внутренней синхронизации - генератор скорости связи

Внутренняя синхронизация используется для асинхронного и ведущего синхронного режимов работы. Описание в данном параграфе опирается на рис. 62. Регистр генератора скорости связи (UBRR) и связанный с ним вычитающий счетчик функционируют как программируемый предделитель или генератор скорости связи. Вычитающий счетчик тактируется системной синхронизацией (fosc) и перезагружается значением из регистра UBRR всякий раз при достижении нулевого значения или после записи регистра UBRRL. Тактовый сигнал генерируется всякий раз при достижении счетчиком нулевого значения. Данный тактовый сигнал является тактовым выходом генератора скорости связи (= fosc/(UBRR+1)). Передатчик делит частоту генератора скорости связи на 2, 8 или 16 в зависимости от режима работы. Модули обнаружения синхронизации и данных приемника подключены непосредственно к тактовому выходу генератора скорости связи. Однако, цифровой автомат модулей обнаружения используют 2, 8 или 16 состояний в зависимости от режима, задаваемого битами UMSEL, U2X и DDR_XCK. Таблица 52 содержит выражения для вычисления скорости связи (в битах в секунду) и вычисления значений UBRR для каждого из рабочих режимов при использовании внутренне генерируемого тактового источника.

Таблица 52. Выражения для вычисления установок регистра скорости связи

Режим работы	Выражение для вычисления (1) скорости связи	Выражения для вычисления значения UBRR
Нормальный асинхронный режим (U2X = 0)	$BAUD = \frac{f_{osc}}{16(UBRR+1)}$	$UBRR = \frac{f_{osc}}{16BAUD} - 1$
Асинхронный режим с удвоением скорости (U2X=1)	$BAUD = \frac{f_{osc}}{8(UBRR+1)}$	$UBRR = \frac{f_{osc}}{8BAUD} - 1$
Синхронный ведущим режим	$BAUD = \frac{f_{osc}}{2(UBRR+1)}$	$UBRR = \frac{f_{osc}}{2BAUD} - 1$

Прим. 1: Скорость связи представлена в битах в секунду (бод).

BAUD - скорость связи (в битах в секунду, бод)
fOSC - частота синхронизации системного генератора
UBRR - Содержимое регистров UBRRH и UBRRL, (0 … 4095)
Примеры значений UBRR для некоторых частот системной синхронизации представлены в таблице 60. (см. страницу 159).

Работа с удвоением скорости связи (U2X)

Скорость передачи данных может быть удвоена, если установить бит U2X в регистре UCSRA. Установка данного бита оказывает действие только в асинхронном режиме.

При использовании синхронного режима необходимо установить нулевое значение данного бита. Установка данного бита приводит к уменьшению коэффициента деления частоты генератора скорости связи с 16 до 8, тем самым удваивая скорость асинхронной связи. Однако следует обратить внимание, что в этом случае приемник сокращает количество выборок с 16 до 8 при обнаружении синхронизации и данных, поэтому, при использовании данного режима необходимо использовать более точные установки скорости связи и более стабильный тактовый источник. Для передатчика удвоение скорости не связано с какими-либо ограничениями.

Внешняя синхронизация

Внешняя синхронизация используется в синхронном подчиненном режиме работы (см. рис. 62). Во избежание возможности возникновения метастабильности вход внешней синхронизации с вывода XCK связан с регистром синхронизации. Выход регистра синхронизации проходит через детектор фронтов, а только затем используется приемником и передатчиком. На данный процесс затрачивается два такта синхронизации ЦПУ и, поэтому, максимальная частота внешней синхронизации на выводе XCK ограничивается следующим выражением:

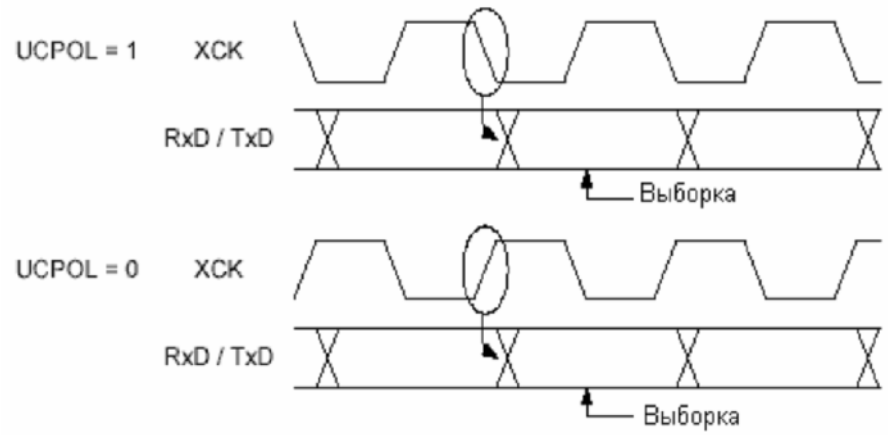
$$f_{XCK} < \frac{f_{OSC}}{4}$$

Обратите внимание, что частота fosc зависит от стабильности системного источника синхронизации. В связи с этим рекомендуется учесть некоторый запас для предотвращения возможности потери данных из-за колебаний частоты.

Режим синхронной связи

Если используется режим синхронной связи (UMSEL = 1), то вывод XCK используется или как вход синхронизации (подчиненный режим) или как выход синхронизации (ведущий режим). Зависимость между тактовыми фронтами и выборкой данных или изменением данных одна и та же. Основной принцип работы заключается в том, что выборка вводимых данных (на RxD) осуществляется фронтом XCK, который противоположен фронту, по которому происходит изменение выходных данных (на TxD).

Рисунок 63. Временная диаграмма для синхронного режима XCK



Бит UCPOL регистра UCRSC выбирает какой фронт XCK используется для выборки данных, а какой для изменения данных. На рисунке 63 показано, что при UCPOL=0 изменение данных происходит по нарастающему фронту XCK, а выборка по падающему фронту XCK. Если установлен бит UCPOL, то изменение данных происходит по падающему фронту XCK, а выборка по нарастающему фронту XCK.

Форматы посылки

Последовательная посылка состоит из бит данных, бит синхронизации (старт и стоп-биты), а также опционального бита паритета для поиска ошибок. УСАПП поддерживает все 30 комбинаций следующих форматов посылок:

- 1 старт-бит
- 5, 6, 7, 8 или 9 бит данных
- без паритета, с битом четности, с битом нечетности
- 1 или 2 стоп-бита

Посылка начинается со старт-бита, а за ним следует передача бит данных, начиная с самого младшего разряда. Затем следует передача остальных бит данных (макс. число бит данных 9), которая заканчивается передачей старшего разряда данных. Если разрешена функция контроля паритета, то сразу после бит данных передается бит паритета, а затем стоп-биты. После завершения передачи посылки имеется возможность либо передавать следующую посылку либо перевести линию связи в состояние ожидания (высокий уровень). Рисунок 64 иллюстрирует возможность сочетания форматов посылки. Наличие прямоугольной скобки указывает на опциональность данного формата посылки.

Рисунок 64. Форматы посылки

· St - Старт-бит имеет всегда низкий уровень.
· 0…8 - Номер бита данных.
· P - бит паритета: четность или нечетность.

· Sp1 - Стоп-бит имеет всегда высокий уровень.
· IDLE - состояние ожидания, в котором приостановлена передача на RxD или TxD. В состоянии
ожидания на линии должен быть высокий уровень.

Формат посылки, который используется УСАПП, задается битами UCSZ2:0, UPM1:0 и USBS в регистрах UCSRB и UCSRC. Приемник и передатчик используют одни и те же установки форматов. Обратите внимание, что изменение установок любого из этих бит может привести к повреждению текущего сеанса связи, как для приемника, так и для передатчика.

Биты выбора длины передаваемых данных (UCSZ2:0) определяют из скольки бит данных состоит посылка. Биты режима паритета УСАПП (UPM1:0) разрешают передачу/контроля бита паритета и устанавливают тип паритета: четность, нечетность. Выбрать один или два стоп-бита позволяет бит выбора стоп-бита УСАПП (USBS). Приемник игнорирует второй стоп-бит. Флаг ошибки посылки FE позволяет выявить ситуацию, когда первый стоп-бит равен 0.

Вычисление бита паритета

Бит паритета вычисляется путем выполнения логической операции исключающего ИЛИ над всеми битами данных. Если используется нечетность, то результат этой операции инвертируется. Сказанное отражено в следующих выражениях:

$$P_{ЧЕТН} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$

$$P_{НЕЧЕТН} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

где
РЧЕТН - бит четного паритета;
РНЕЧЕТН - бит нечетного паритета;
dn - n-ый бит данных в посылке.
После разрешения, бит паритета передается между последним битом данных и первым стоп- битом.

Инициализация УСАПП

Перед началом сеанса связи необходимо выполнить инициализацию УСАПП. Процесс инициализации обычно состоит из установки скорости связи, задания формата посылки и разрешения работы передатчика и приемника. Если используется управление связью по прерываниям, то во время инициализации необходимо, чтобы был сброшен флаг общего разрешения прерываний (т.е. необходимо запретить все прерывания).

Если необходимо выполнить повторную инициализацию УСАПП, например, для изменения скорости связи или формата посылки, то необходимо убедиться, чтобы во время инициализации передача была приостановлена. Флаг TXC может использоваться для проверки завершения работы передатчика, а флаг RXC - для проверки отсутствия в приемном буфере несчитанных данных. Обратите внимание, что при использовании флага TXC он должен сбрасываться программно перед началом каждой передачи (перед записью в UDR).

В следующих примерах показаны функции для простой инициализации УСАПП на Ассемблере и Си. В примерах предполагается, что используются управление связью по опросу флагов состояния (не по прерываниям) и фиксированный формат посылки. Скорость связи выступает как параметр функции. Для примера на ассемблере предполагается, что параметр скорости связи записан перед вызовом функции в регистры r17:r16. Когда функция пишет в Регистр UCSRC, бит URSEL (MSB) должен быть установлен из-за совместного использования расположения ввода-вывода UBRRH и UCSRC.

Пример кода на Ассемблере (1)

```
USART_Init:
; Установка скорости связи
out UBRRH, r17
174
out UBRRL, r16
; Разрешение работы приемника и передатчика
ldi r16, (1<<RXEN)|(1<<TXEN)
out UCSRB,r16
; Установка формата посылки: 8 бит данных, 2стоп-бита
ldi r16, (1<<URSEL)|(1<<USBS)|(3<<UCSZ0)
out UCSRC,r16
ret
```

Пример кода на Си (1)

```
#define FOSC 1843200// Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
void main( void )
{
...
USART_Init ( MYUBRR );
...
}
void USART_Init( unsigned int ubrr)
{
/* Установка скорости связи */
UBRRH = (unsigned char)(ubrr>>8);
UBRRL = (unsigned char)ubrr;
/* Разрешение работы передатчика и приемника */
UCSRB = (1<<RXEN)|(1<<TXEN);
/* Установка формата посылки: 8 бит данных, 2 стоп-бита */
UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
```

Прим.: 1. См. “О Примерах кода” на странице 8.

Более совершенные процедуры инициализации могут использовать расширенный интерфейс функции, где в качестве параметров выступают, например, формат посылки, отключение прерываний и т.д. Однако, в большинстве приложений используется фиксированные установки скорости связи и управляющих регистров, поэтому, для них представленные примеры могут быть непосредственно включены в основную программу или к процедурам инициализации других модулей ввода-вывода.

Передача данных - Передатчик УСАПП

Работа передатчика УСАПП разрешается путем установки бита разрешения передачи (TXEN) в регистре UCSRB. После разрешения, функция вывода TxD как обычного порта заменяется на функцию выхода последовательной передачи данных. Скорость связи, режим работы и формат посылки должны быть установлены однократно перед началом какой-либо передачи. Если используется синхронная работа, то функция вывода XCK также заменяется на альтернативную - синхронизация передачи.

Передача посылок с 5…8 битами данных

Начало передачи инициируется записью передаваемых данных в буфер передатчика. ЦПУ может загрузить буфер передатчика путем записи в регистр UDR, расположенный в памяти ввода- вывода. Буферизованные данные в буфере передатчика будут перемещены в сдвиговый регистр, после того как он будет готов к отправке новой посылки. Запись в сдвиговый регистр новых данных происходит в состоянии ожидания (когда передача завершена) или сразу после завершения передачи последнего стоп-бита предыдущей посылки. Если в сдвиговый регистр записаны новые данные, то начинается передача одной посылки на скорости, определенной в регистре скорости связи, битом U2X или XCK в зависимости от выбранного режима работы.

В следующих примерах представлены простые функции передачи через УСАПП, использующие опрос флага освобождения регистра данных (UDRE). Если используется посылка с менее 8 бит данных, то старшие биты записанные в UDR игнорируются. Перед вызовом данной функции должна быть выполнена инициализация УСАПП. Для кода на Ассемблере предполагается, что передаваемые данные записаны в регистр R16 перед вызовом процедуры.

Пример кода на Ассемблере (1)

```
USART_Transmit:
; Ожидание освобождения буфера передатчика
sbis UCSRA,UDRE
rjmp USART_Transmit
; Помещение данных (r16) в буфер, отправка данных
out UDR,r16
ret
```

Пример кода на Си (1)

```
void USART_Transmit( unsigned char data )
{
/* Ожидание освобождения буфера передатчика */
while ( !( UCSRA & (1<<UDRE)) );
/* Помещение данных в буфер, отправка данных */
UDR = data;
}
```

Прим.: 1. См. “О Примерах кода” на странице 8.

Перед загрузкой новых данных для передачи в данной функции осуществляется ожидание освобождения буфера передатчика путем опроса флага UDRE.
Если регистр данных использует пустое прерывание, подпрограмма прерывания, пишет данные в буфер.

Отправка посылок с 9 битами данных

Если необходимо передавать 9 бит данных (UCSZ = 7), то 9-ый бит данных должен быть записан в бит TXB8 регистра UCSRB, перед тем как младший байт будет записан в UDR. В следующих примерах показаны функции для передачи 9 бит данных. Для кода на Ассемблере предполагается, что отправляемые данные предварительно записаны в регистры R17:R16.

Пример кода на Ассемблере (1)

```
USART_Transmit:
; Ожидание освобождения буфера передатчика
sbis UCSRA,UDRE
rjmp USART_Transmit
; Копирование 9-го бита из r17 в TXB8
cbi UCSRB,TXB8
sbrc r17,0
sbi UCSRB,TXB8
; Помещение мл. байта данных (r16) в буфер, отправка данных
out UDR,r16
ret
```

Пример кода на Си (1)

```
void USART_Transmit( unsigned int data )
{
/* Ожидание освобождения буфера передатчика */
while ( !( UCSRA & (1<<UDRE)) );
/* Копирование 9-го бита в TXB8 */
UCSRB &= ~(1<<TXB8);
if ( data & 0x0100 )
UCSRB |= (1<<TXB8);
/* Помещение данных в буфер, отправка данных */
UDR = data;
}
```

Прим.1: Данные функции записаны как функции общего назначения. Они оптимизированы под статическое содержимое UCSRB. Т.е. когда после инициализации в регистре UCSRB изменяется только бит TXB8.

9-ый бит данных может использоваться для индикации адреса посылки в многопроцессорном режиме связи или в других протоколах.

Флаги и прерывания передатчика

Передатчик УСАПП имеет два флага, которые индицируют его состояние: флаг освобождения регистра данных УСАПП (UDRE) и флаг завершения передачи (TXC). Оба флага могут использоваться для генерации прерываний. Флаг освобождения регистра данных (UDRE) индицирует готовность буфера передатчика принять новые данные. Данный бит устанавливается при освобождении передающего буфера и сбрасывается, когда буфер передатчика содержит данные для передачи, но которые еще не были переданы в сдвиговый регистр. Для совместимости с будущими микроконтроллерами в данный бит необходимо записывать ноль во время записи в регистр UCSRA.

Если записать лог. 1 в бит разрешения прерывания по освобождению регистра данных (UDRIE) в регистре UCSRB, то прерывание по освобождению регистра данных УСАПП будет выполняться всякий раз, когда устанавливается бит UDRE (с учетом того, что общие прерывания разрешены).
UDRE сбрасывается при записи в UDR.

Если используется управление связью по прерываниям, то в процедуре обработки прерывания по освобождению регистра данных необходимо или записать новые данные в регистр UDR для сброса флага UDRE или выключить прерывание по освобождению регистра данных. В противном случае при выходе из процедуры обработки прерывания сразу возникнет новое прерывание.

Флаг завершения передачи (TXC) принимает единичное значение, если вся посылка в сдвиговом регистре передатчика была полностью сдвинута и в буфере передатчика отсутствуют новые данные для передачи. Флаг TXC автоматически сбрасывается при переходе на вектор обработки прерывания по завершению передачи или программно сбрасывается путем записи в него лог. 1. Флаг TXC полезно использовать при организации полудуплексной связи (например, стандарт RS485), где имеется необходимость перевода передающей стороны в режим приема после завершения передачи, тем самым достигая освобождения шины.

Если разрешено прерывание по завершению передачи (TXCIE=1) в регистре UCSRB, то прерывание по завершению передачи УСАПП выполняется всякий раз, когда флаг TXC принимает единичное состояние (с учетом, что активно общее разрешение прерываний). При переходе на вектор обработки прерывания по завершении передачи УСАПП нет необходимости сбрасывать флаг TXC, т.к. это происходит автоматически.

Генератор паритета

Генератор паритета вычисляет бит паритета для включения в состав последовательной посылки. Если бит паритета установлен (UPM1 = 1), то управляющая логика передатчика вставляет бит паритета между последним битом данным и первым стоп-битом в отправляемой посылке.

Отключение передатчика

Отключение передатчика после сброса TXEN наступит только тогда, когда завершится текущая и ожидаемая передача, т.е. когда в сдвиговом регистре передатчика и буфере передатчика не будет данных для передачи. После отключения передатчика отменяется альтернативное назначение вывода TxD.

Прием данных - Приемник УСАПП

Работа приемника УСАПП разрешается, если записать лог. 1 в бит разрешения работы приемника (RXEN) в регистре UCSRB. После разрешения работы приемника обычное назначение вывода RxD заменяется на альтернативное: вход последовательного ввода данных приемника УСАПП. Скорость связи, режим работы и формат посылки должны быть установлены однократно перед началом выполнения приема данных. Если используется синхронная работа, то вывод XCK будет использоваться для синхронизации связи.

Прием посылок с 5…8 битами данных

Приемник начинает прием данных только после определения действительного старт-бита. Выборка следующих за старт-битом бит данных происходит с частотой равной скорости связи или частотой сигнала XCK и размещается в сдвиговом регистре приемника. Второй стоп-бит приемником игнорируется. После получения первого стоп-бита, т.е. когда последовательная посылка полностью принята и находится в сдвиговом регистре приемника, содержимое сдвигового регистра перемещается в приемный буфер. Приемный буфер считывается при чтении регистра ввода-вывода UDR.

В следующих примерах приведены простые функции для организации приема данных, который основан на опросе состояния флага завершения приема (RXC). Если используется формат посылки с числом бит менее 8, то после считывания содержимого UDR старшие неиспользуемые разряды будет обнулены. Перед вызовом данных функций должна быть выполнена инициализация УСАПП.

Пример кода на Ассемблере (1)

```
USART_Receive:
; Ожидание окончания приема данных
sbis UCSRA, RXC
rjmp USART_Receive
; Загрузка принятых данных из буфера
in r16, UDR
ret
```

Пример кода на Си (1)

```
unsigned char USART_Receive( void )
{
/* Ожидание окончания приема данных */
while ( !(UCSRA & (1<<RXC)) );
/* Загрузка принятых данных из буфера */
return UDR;
}
```

Прим.: 1. См. “О Примерах кода” на странице 8

Прежде чем считывается содержимое буфера приемника, в функциях выполняется ожидание появления данных в этом буфере путем опроса флага RXC.

Прием посылок с 9 битами данных

Если используется передача 9 бит данных (UCSZ=7), то непосредственно перед чтением младшего байта из UDR необходимо считать значение 9-го бита данных RXB8 в регистре UCSRB. Данное правило также относится к флагам статуса FE, DOR и UPE: сначала опрашиваем состояние UCSRA, а только затем считываем данные из UDR. Данное ограничение связано с тем, что принимаемые данные буферизуются вместе с флагами статуса, поэтому, считывание регистра UDR приводит к изменению состояния приемного буфера FIFO и, следовательно, связанные со считанными данными биты TXB8, FE, DOR и UPE будут потеряны.

Ниже приведены примеры функций для организации приема 9 бит данных с флагами статуса.

Пример кода на Ассемблере (1)

```
USART_Receive:
; Ожидание окончания приема данных
sbis UCSRA, RXC
rjmp USART_Receive
; Опрос статусных бит и 9-го бита данных перед чтением данных из буфера
in r18, UCSRA
in r17, UCSRB
in r16, UDR
; Если ошибка, то возврат -1
andi r18,(1<<FE)|(1<<DOR)|(1<<UPE)
breq USART_ReceiveNoError
ldi r17, HIGH(-1)
ldi r16, LOW(-1)
USART_ReceiveNoError:
; Выделение 9-го бита данных перед выходом
lsr r17
andi r17, 0x01
ret
```

Пример кода на Си (1)

```
unsigned int USART_Receive( void )
{
unsigned char status, resh, resl;
/* Ожидание окончания приема данных */
while ( !(UCSRA & (1<<RXC)) );
/* Опрос статусных бит и 9-го бита данных перед чтением данных из буфера */
status = UCSRA;
resh = UCSRB;
resl = UDR;
/* Если ошибка, то возврат -1 */
if ( status & (1<<FE)|(1<<DOR)|(1<<UPE) )
return -1;
/* Выделение 9-го бита данных перед выходом */
resh = (resh >> 1) & 0x01;
return ((resh << 8) | resl);
}
```

Прим.: 1. См. “О Примерах кода” на странице 8

В данных функциях считываются все регистры ввода-вывода в файл регистров перед выполнением каких-либо вычислений. Такой подход позволяет наиболее оптимально использовать заполняемость буфера, т.к. буфер становится свободным для приема новых данных, как только это станет возможным.

Флаг и прерывание по завершению приема

Приемник УСАПП имеет один флаг, который индицирует состояние приемника.

Флаг завершения приема (RXC) сигнализирует о наличии несчитанных данных в приемном буфере. Данный флаг равен 1, если имеются несчитанные данные, и равен 0, если буфер приемника свободен (т.е. не содержит каких-либо несчитанных данных). Если приемник отключается (RXEN = 0), то приемный буфер будет сброшен и флаг RXC примет нулевое значение.

Если установлен бит разрешения прерывания по завершению приема (RXCIE) в регистре UCSRB, то при установке флага RXC программа переход не вектор обработки данного прерывания (при условии, что активно общее разрешение прерываний). Если используется организация связи с управлением по прерываниям, то при выполнении процедуры обработки запроса на прерывание по завершению приема необходимо считать данные из UDR, чтобы сбросить флаг RXC. В противном случае новое прерывание возникнет сразу после выхода из текущего.

Флаги ошибок приемника

Приемник УСАПП имеет три флага ошибок: ошибка посылки (кадра) FE, переполнение данных DOR и ошибка паритета UPE. Данные флаги входят в состав регистра UCSRA. Общим свойством данных флагов является то, что они хранятся в приемном буфере вместе с той посылкой данных, для которой они отражают состояние ошибок. С учетом этого, необходимо следить, чтобы флаги ошибок считывались из регистра UCSRA перед чтением данных из приемного буфера (UDR), т.к. после чтения из UDR изменяется состояние буфера. Другим сходством флагов ошибок является невозможность программно повлиять на их состояние. Однако, в целях совместимости с УСАПП последующих микроконтроллеров во время записи регистра UCSRA в позициях флагов ошибок необходимо указывать нулевые значения. Ни один из флагов ошибок не может вызвать прерывание.

Флаг ошибки посылки (кадра) FE индицирует состояние первого стоп-бита сохраненной в приемном буфере посылки. Флаг FE равен 0, если стоп-бит имел корректное значение (лог. 1), и равен 1, если некорректное, т.е. 0. Данный флаг может использоваться для выявления условия разсинхронизации, обрыва связи и манипуляции над протоколом связи. Флаг FE не изменяется при установке бита USBS в регистре UCSRC, т.к. приемник игнорирует все стоп-биты за исключением первого. Для совместимости с последующими микроконтроллерами в позиции данного бита необходимо указывать 0 во время записи в регистр UCSRA.

Флаг переполнения данных (DOR) сигнализирует о потере данных из-за переполнения приемного буфера. Переполнение данных возникает, если приемный буфер заполнен (две посылки), в сдвиговом регистре ожидает считывания только что принятая посылка и обнаружен новый старт-бит. Если флаг DOR установлен, то значит одна или более последовательных посылок потеряны между последним и следующим считанными значениями из UDR. Для совместимости с будущими микроконтроллерами в позицию данного бита необходимо всегда записывать лог.0 во время записи в регистр UCSRA. Флаг DOR сбрасывается, если принятая посылка была успешно перемещена из сдвигового регистра в приемный буфер.

Флаг ошибки паритета (UPE) сигнализирует, что во время приема посылки была обнаружена ошибка паритета. Если контроль паритета отключен, то данный флаг всегда имеет нулевое значение. Для совместимости с новыми разработками микроконтроллеров в позицию данного бита необходимо всегда записывать 0 во время записи в регистр UCSRA. См. также "Вычисление бита паритета" на странице 138 и "Устройство проверки паритета" на странице 147.

Устройство проверки паритета

Устройство проверки паритета становится активным после установки бита режима паритета УСАПП UPM1. Тип контроля паритета: четность или нечетность задается битом UPM0. После активизации устройство проверки паритета вычисляет паритет принятых данных и сравнивает полученное значение с принятым вместе с этими данными в одной посылке битом паритета. Результат сравнения запоминается в приемном буфере вместе с принятыми данными и стоп- битом. Флаг ошибки паритета UPE может быть считан программно тогда, когда в посылке имеется ошибка паритета.

Бит UPE устанавливается, если в посылке, которая может быть считана из приемного буфера, имеется ошибка паритета и во время приема этой посылки был разрешен контроль паритета (UPM1 = 1). Данный бит должен быть опрошен до считывания буфера приемника (UDR).

Отключение приемника

В отличие от передатчика, отключение приемника происходит незамедлительно. При этом, принимаемые данные будут потеряны. После отключения (т.е. когда RXEN =0) приемник далее не поддерживает альтернативные настройки вывода порта RxD. Приемный буфер FIFO сбрасывается после отключения приемника, следовательно, оставшиеся в нем данные будут потеряны.

Сброс приемного буфера

Приемный буфер FIFO сбрасывается после отключения приемника, т.е. полностью освобождается от своего содержимого. Несчитанные данные будут потеряны. Если буфер необходимо очистить в процессе нормальной работы, например, при возникновении ошибок, то необходимо считывать содержимое UDR пока не очиститься флаг RXC. В следующем примере показано как очистить буфер приемника.

Пример кода на Ассемблере (1)

```
USART_Flush:
sbis UCSRA, RXC
ret
in r16, UDR
rjmp USART_Flush
```

Пример кода на Си (1)

```
void USART_Flush( void )
{
unsigned char dummy;
while ( UCSRA & (1<<RXC) ) dummy = UDR;
}
```

Прим.: 1. См. “О Примерах кода” на странице 8

Асинхронный прием данных

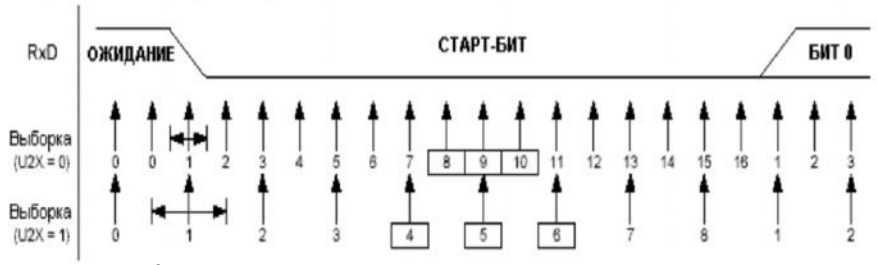
УСАПП содержит блоки обнаружения данных и синхронизации для управления асинхронным приемом данных. Логика обнаружения синхронизации используется для синхронизации с внутренним генератором скорости связи для обеспечения возможности ввода последовательной посылки с выв. RxD. Логика обнаружения данных осуществляет выборку и фильтрацию (ФНЧ) каждого входящего бита данных, тем самым увеличивая помехоустойчивость приемника. Рабочий диапазон асинхронного приема определяется точностью встроенного генератора скорости связи, точностью скорости входящей посылки и размером посылки (количество бит).

Асинхронный поиск синхронизации

Логика обнаружения синхронизации синхронизирует во времени работу приемника с входящей последовательной посылкой. На рис. 65 иллюстрируется процесс поиска старт-бита во входящей посылке. Частота выборок в 16 раз выше скорости связи для нормального режима и 8 раз выше для режима удвоения скорости. Горизонтальные стрелки иллюстрируют возможный уход синхронизации в процессе выборки.

Обратите внимание на более высокую разсинхронизацию во времени при использовании режима удвоения скорости (U2X = 1). Выборки, обозначенные номером 0, соответствуют состоянию ожидания на линии RxD (т.е. при неактивной связи).

Рисунок 65. Выборка старт-бита



Если логика обнаружения синхронизации определяет переход из высокого (состояние ожидания) к низкому (старт) состоянию на линии RxD, то инициируется последовательность действий по обнаружению старт-бита. Примем, что выборка 1 означает первая выборка с нулевым значением. Тогда по выборкам 8, 9, 10 в нормальном режиме и выборкам 4, 5, 6 в режиме удвоения скорости определяется действительность старт-бита (на рисунке эти выборки помещены в рамку). Если две или более из этих выборок имеют единичное состояние (принцип мажоритарного голосования), то старт-бит отклоняется как ложный, а приемник продолжит поиск следующего перехода из 1 в 0. Однако если определен действительный старт-бит, то логика обнаружения синхронизации оказывается засинхронизированной, после чего вступит в силу логика обнаружения данных. Процесс синхронизации повторяется для каждого старт-бита.

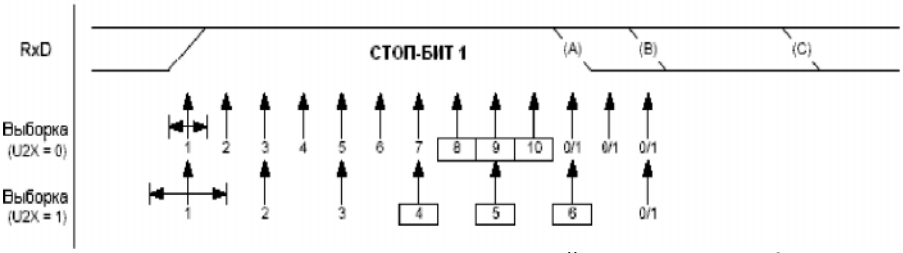
Асинхронный поиск данных

После обнаружения старт-бита начинает работу логика обнаружения данных. Блок обнаружения данных использует цифровой автомат с 16 состояниями в нормальном режиме работы и с 8 состояниями в режиме удвоения скорости. На рисунке 66 показана выборка бит данных и бита паритета. Для каждой выборки указан номер, который соответствует номеру состояния цифрового автомата блока обнаружения данных.

Рисунок 66. Выборка данных и бит паритета

Определение логического уровня принимаемого бита данных происходит с помощью мажоритарного голосования по трем выборкам, расположенных по центру принятого бита. Центральные выборки выделены на рисунке путем размещения их в рамке. Процесс мажоритарного голосования состоит в следующем: если две или все три выборки имеют высокие уровни, то принятый бит фиксируется как лог. 1. Если две или три выборки имеют низкие уровни, то принятый бит фиксируется как лог. 0. Процесс мажоритарного голосования, по сути, представляет собой фильтр низких частот для входящего сигнала с вывода RxD. Процесс обнаружения повторяется до полного завершения приема посылки, в т.ч. первый стоп-бит. Обратите внимание, что приемник определяет только первый стоп-бит посылки, а второй игнорируется. На рисунке 67 отображен процесс выборки стоп-бита и начальный момент возможности обнаружения старт-бита следующей посылки.

Рисунок 67. Выборка стоп-бита и следующего старт-бита



Принцип мажоритарного голосования, рассмотренный на примере стоп-бита, аналогично распространяется и на другие биты в посылке. Если обнаруженный стоп-бит имеет нулевое значение, то устанавливается флаг ошибки посылки FE.

Новое изменение из 1 в 0 будет воспринято как стоп-бит новой посылки, если это изменение произошло после выборки последнего бита, используемого при мажоритарном голосовании. Для режима с нормальной скоростью первая выборка с низким уровнем может находится в позиции, обозначенной A на рисунке 85. Для режима удвоения скорости появление низкого уровня допускается позже (точка B). Точка C соответствует полному завершению передачи стоп-бита. Использование раннего обнаружения старт-бита влияет на рабочий диапазон приемника (допустимое расхождение частот при фиксированном формате посылки).

Рабочий диапазон асинхронной связи

Рабочий диапазон приемника зависит от расхождения между внутренне генерируемой скоростью связи и скоростью принимаемых бит. Если передатчик отправляет посылки на более высокой или более низкой скорости или внутренне-генерируемая скорость связи приемника не соответствует основной частоте (см. табл. 53), то приемник окажется неспособным засинхронизировать посылку по отношению к старт-биту.

Следующие выражения могут использоваться для вычисления отношения скорости принимаемых данных и внутренней скорости приемника:

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_M}$$

Где

D - сумма количества передаваемых бит данных и бит паритета, D = 5…10;
S - количество выборок в секунду. S = 16/8 в режиме нормальной/удвоенной скорости;
SF - Номер первой выборки используемой для мажоритарного голосования. SF = 8/4 в режиме нормальной/удвоенной скорости;
SM - Номер центральной выборки используемой при мажоритарном голосовании. SM = 9/5 в режиме нормальной/удвоенной скорости;
Rмин - отношение наименьшей скорости принимаемых данных к скорости приемника;
Rмакс - отношение набольшей скорости принимаемых данных к скорости приемника.

В таблицах 53 и 54 приведен список максимальных допустимых погрешностей при генерации скорости приемника. Обратите внимание, что режим нормальной скорости устойчив к более широким изменениям скорости связи.

Таблица 53. Рекомендуемая максимальная погрешность генерации скорости связи приемника в режиме нормальной скорости (U2X = 0)

D (кол. бит данных и паритета)	Rмин, %	Rмакс, %	Общая макс. погрешность, %	Рекомендуемая максимальная погрешность приемника, %
5	93,20	106,67	+6.67/-6.8	± 3.0
6	94,12	105,79	+5.79/-5.88	± 2.5
7	94,81	105,11	+5.11/-5.19	± 2.0
8	95,36	104,58	+4.58/-4.54	± 2.0
9	95,81	104,14	+4.14/-4.19	± 1.5
10	96,17	103,78	+3.78/-3.83	± 1.5

Таблица 54. Рекомендуемая максимальная погрешность генерации скорости связи приемника в режиме удвоенной скорости (U2X = 1)

D (кол. бит данных и паритета)	$R_{мин}$, %	$R_{макс}$, %	Общая макс. погрешность, %	Рекомендуемая максимальная погрешность приемника, %
5	94,12	105,66	+5.66/-5.88	± 2.5
6	94,92	104,92	+4.92/-5.08	± 2.0
7	95,52	104,35	+4.35/-4.48	± 1.5
8	96,00	103,90	+3.90/-4.00	± 1.5
9	96,39	103,53	+3.53/-3.61	± 1.5
10	96,70	103,23	+3.23/-3.30	± 1.0

Рекомендуемая погрешность генератора скорости связи приемника была выбрана исходя из того, что передатчик и приемник в совокупности определяют общую максимальную погрешность.

Имеется два возможных источника влияния на погрешность скорости связи приемника. Системная синхронизация (XTAL) приемника всегда имеет некоторую нестабильность в зависимости от напряжения питания и температуры. При использовании кварцевого резонатора для генерации системной синхронизации как правило не возникает проблем, но при использовании керамических резонаторов частота синхронизации может изменяться более чем на 2% в зависимости характеристик выбранного резонатора. Второй источник влияния на погрешность является более управляемым. Требуемую скорость связи не всегда удается получить путем деления частоты синхронизации на целое число. В этом случае необходимо выбрать такое значение UBRR, которое обеспечивает минимально возможную погрешность результирующей частоты.

Многопроцессорный режим связи

Установка бита многопроцессорного режима связи MPCM в регистре UCSRA активизирует функцию фильтрации входящих посылок приемником УСАПП. Посылки, которые не содержат информации об адресе игнорируются и не помещаются в приемный буфер. Это позволяет существенно уменьшить количество входящих посылок подлежащих обработке ЦПУ в многопроцессорных системах, связь между процессорами в которых организована через одну последовательную шину. Значение бита MPCM не оказывает ни какого влияния на работу передатчика, но при этом передатчик должен быть использован иначе, если используется режим многопроцессорной связи.

Если приемник настраивается на прием посылок с 5…8 битами данных, то первый стоп-бит позволяет отличить назначение принятых данных: адрес или данные. Если приемник настроен на прием 9 бит данных, то значение 9-го бита (RXB8) используется для идентификации адреса или данных. Если идентификатор типа посылки (первый стоп-бит или 9-ый бит данных) равен 1, то в посылке содержится адрес. В противном случае в посылке переданы данные.

Режим многопроцессорной связи позволяет нескольким подчиненным микроконтроллерам принимать данные от одного ведущего. При этом подчиненные микроконтроллеры по первой адресной посылке определяют к какому микроконтроллеру адресуется ведущий. Если один из подчиненных микроконтроллеров обнаруживает свой адрес, то следующие посылки данных он будет принимать в нормальном режиме, а остальные подчиненные микроконтроллеры эти данные игнорируют до тех пор, пока не будет обнаружена следующая адресная посылка.

Использование MPCM

Если микроконтроллер действует как ведущий, то он может использовать 9-битный формат данных в посылке (UCSZ = 7). 9-ый бит данных (TXB8) устанавливается при передаче адресной посылки (TXB8 = 1) и сбрасывается при передаче посылки данных (TXB = 0). В этом случае подчиненные микроконтроллеры также должны устанавливать 9-битный формат.

Для обмена данными в многопроцессорном режиме связи необходимо использовать следующие процедуры:
1. Все подчиненные микроконтроллеры переводятся в многопроцессорный режим связи (MPCM =1 в UCSRA).
2. Ведущий МК отправляет адресную посылку, а все подчиненные принимают и считывают эту посылку. В подчиненных МК флаг RXC в регистре UCSRA устанавливается как обычно.
3. Каждый подчиненный МК считывает регистр UDR и определяет к кому адресуется ведущий МК. Адресуемый МК должен очистить бит MPCM в UCSRA, в противном случае он ожидает следующего адресного байта и сохраняет установки MPCM.
4. Адресуемый МК принимает все данные до следующей адресной посылки. Другие подчиненные МК, у которых бит MPCM остался установленным будут игнорировать посылки данных.
5. После приема адресуемым МК последней посылки данных устанавливается бит MPCM и ожидается прием новой адресной посылки от ведущего МК. Далее процесс повторяется с пункта 2.

Использование 5..8-разрядных форматов данных возможно, но не удобно, т.к. приемник должен переключаться между n и n+1 форматами посылки. Это делает затруднительной полнодуплексную связь, т.к. передатчик и приемник используют общие установки формата. При использовании 5…8- разр. данных в посылке передатчик должен использовать два стоп-бита, т.к. первый стоп-бит будет задействован для индикации типа посылки.

Не пользуйтесь инструкциями "чтение-модификация-запись" (SBI и CBI) для установки или сброса бита MPCM. Бит MPCM находится в одной ячейке с флагом TXC, поэтому, последний может быть случайно сброшен при выполнении инструкций SBI или CBI.

Доступ к регистрам UBRRH/UCSRC

Регистр UBRRH совместно использует то же самое расположение ввода-вывода как Регистр UCSRC. Поэтому некоторое специальное рассмотрение должно быть, чтобы получить доступ к этому расположению ввода-вывода.

Доступ Записи

Делая доступ записи этого расположения ввода-вывода, нужно записать в бит (URSEL) регистра USART , нужное значение, который из двух регистров будет записан. Если URSEL будет нулем во время работы записи, то значение UBRRH будет обновлено. Если URSEL будет один, то установка UCSRC будет обновлена. Следующие примеры кода показывают, как получить доступ к двум регистрам.

Assembly Code Examples(1)

```
...
; устанавливаем UBRRH в 2
ldi r16,0x02
out UBRRH,r16
...
; устанавливаем USBS и  UCSZ1 bit в 1, и
; остальные биты в 0.
ldi r16,(1<<URSEL)|(1<<USBS)|(1<<UCSZ1)
out UCSRC,r16
...
```

C Code Examples(1)

```
...
/* Set UBRRH to 2 */
UBRRH = 0x02;
...
/* Set the USBS and the UCSZ1 bit to one, and */
/* the remaining bits to zero. */
UCSRC = (1<<URSEL)|(1<<USBS)|(1<<UCSZ1);
...
```

Прим.: 1. См. “О Примерах кода” на странице 8

Примеры кода иллюстрируют, что доступ записи, двух регистров не затрагивает их совместного использования расположения ввода-вывода.

Доступ чтения

Выполнение доступа чтения к UBRRH или Регистру UCSRC является более сложной задачей. Однако, в большинстве приложений, редко необходимо читать любой из этих регистров.

Доступом чтения управляет синхронизированная последовательность. Чтение расположения ввода-вывода при первом чтении возвращает содержание регистра UBRRH. Если расположение регистра было считано в предыдущем системном такте, то чтение регистра в текущем такте возвратит содержание UCSRC. Отметьте, что синхронизированная последовательность для того, чтобы считать UCSRC является последовательной работой. Прерываниями нужно поэтому управлять (например, отключая прерывания глобально) во время работы чтения.

```
Assembly Code Example
USART_ReadUCSRC:
; Читаем UCSRC
in r16,UBRRH
in r16,UCSRC
ret


C Code Example
unsigned char USART_ReadUCSRC( void )
{
unsigned char ucsrc;
/* Читаем UCSRC */
ucsrc = UBRRH;
ucsrc = UCSRC;
return ucsrc;
}
```

Прим.: 1. См. “О Примерах кода” на странице 8
Пример ассемблерного кода возвращает значение UCSRC в r16.

Чтение содержания UBRRH не является последовательной работой, и поэтому оно может быть считано как обычный регистр.

Описание регистров УСАПП

Регистр данных УСАПП – UDR

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (Read)
	TXB[7:0]								UDR (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Буферные регистры данных передатчика и приемника УСАПП расположены по одному и тому же адресу в области ввода-вывода, обозначенной как регистр данных УСАПП или UDR. Если выполнять запись по адресу регистра UDR, то записываемые данные помещаются в буферный регистр данных передатчика TXB. По аналогии, при чтении регистра UDR извлекается содержимое буферного регистра данных приемника RXB.

При использования 5-, 6- или 7-битных форматов данных передатчик игнорирует, а приемник устанавливает нулевые значения неиспользуемых разрядов.

Запись в буфер передатчика можно выполнять, если установлен флаг UDRE в регистре UCSRA. Данные записанные в UDR при сброшенном флаге UDRE будут игнорированы передатчиком УСАПП. Если выполнена запись в приемный буфер и при этом работа передатчика была разрешена, то после освобождения сдвигового регистра передатчик загрузит в него значение из буферного регистра. После этого выполняется передача данных на выводе TxD. Приемный буфер организован как двухуровневый буфер FIFO (первый пришел - последний вышел). Буфер FIFO изменяет свое состояние, если выполнено чтение из приемного буфера. Вследствие такой организации буфера необходимо следить, чтобы по данному адресу не использовались инструкции "чтение-модификация-запись" (SBI и CBI). Также нужно быть внимательным при использовании инструкций тестирования бита (SBIC и SBIS), т.к. их выполнение может также изменить состояние буфера FIFO.

Регистр A управления и статуса УСАПП – UCSRA

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

• **Bit 7 – RXC:** Флаг завершения приема УСАПП
Данный флаг устанавливается, если в приемном буфере содержатся несчитанные данные и сбрасывается, когда приемный буфер свободен (т.е., не содержит несчитанных данных). Если приемник отключается, то приемный буфер сбрасывается и, следовательно, флаг RXC принимает нулевое значение. Флаг RXC может использоваться для генерации прерывания по завершению приема (см. описание бита RXCIE).

• **Bit 6 – TXC:** Флаг завершения передачи УСАПП
Данный флаг устанавливается, если вся посылка из сдвигового регистра передатчика полностью передана и в передающем буфере UDR нет новых данных для передачи. Флаг TXC автоматически сбрасывается при переходе на вектор прерывания по завершению передачи или сбрасывается программно путем записи лог. 1 в позицию данного бита. Флаг TXC может служить источником для генерации прерывания по завершению передачи (см. также описание бита TXCIE).

• **Bit 5 – UDRE:** Флаг освобождения регистра данных УСАПП
Флаг UDRE индицирует о готовности приемного буфера UDR к приему новых данных. Если UDRE=1, то буфер свободен и, следовательно, готов к записи. Флаг UDRE может служить источником для генерации прерывания по освобождению регистра данных (см. описание бит UDRIE). UDRE устанавливается после сброса, индицируя о готовности передатчика.

• **Bit 4 – FE:** Ошибка посылки
Данный бит устанавливается, если при приеме посылки, находящейся на выходе из приемного буфера, была определена ошибка в структуре посылки. Под ошибкой структуры в данном случае понимается нулевое значение первого стоп-бита в этой посылке. Значение данного бита действительно до чтения содержимого приемного буфера (UDR). Флаг FE принимает нулевое значение, если принятый стоп-бит имел правильное единичное значение. При записи в регистр UCSRA в позиции данного бита необходимо указывать лог. 0.

• **Bit 3 – DOR:** Флаг переполнения данных
Данный бит устанавливает, если выявлено условие переполнения. Переполнение данных возникает, если заполнен приемных буфер (две посылки), новая посылка полностью принята в приемный сдвиговый регистр, а также обнаружен новый старт-бит. Значение данного бита действительно до чтения содержимого приемного буфера (UDR). При записи в регистр UCSRA в позиции данного бита необходимо указывать лог. 0.

• **Bit 2 – PE:** Ошибка паритета
Данный бит устанавливается, если следующая посылка в приемном буфере характеризуется ошибкой паритета, если во время приема этой посылки был разрешен контроль паритета (UPM1 = 1). Данный бит имеет действительное значение до чтения приемного буфера (UDR). При записи в регистр UCSRA в позиции данного бита необходимо указывать лог. 0.

• **Bit 1 – U2X:** Удвоение скорости связи УСАПП

Данный бит оказывает влияние только в асинхронном режиме связи. В синхронном режиме в данный бит необходимо записать лог. 0. Запись в данный бит лог. 1 уменьшает в два раза значение коэффициента деления скорости связи с 16 до 8, тем самым удваивая скорость передачи данных в асинхронном режиме.

• **Bit 0 – MPCM:** Режим многопроцессорной связи
Данный бит разрешает режим многопроцессорной связи. Если в бит MPCM записать лог. 1, то все входящие посылки принимаемые приемником УСАПП будут игнорироваться, если они не содержат адресной информации. Установка бита MPCM не влияет на работу передатчика. Более подробная информация по данному режиму приведена в параграфе "**Многопроцессорный режим связи**" на странице 151.

Регистр B управления и статуса УСАПП – UCSRB

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• **Bit 7 – RXCIE:** Разрешение прерывания по завершению приема
Запись в данный бит лог. 1 разрешает прерывание по флагу RXC. Прерывание по завершению приема УСАПП генерируется, если RXCIE=1, флаг общего разрешения прерываний I=1 (в регистре SREG), а также установлен бит RXC в регистре UCSRA.

• **Bit 6 – TXCIE:** Разрешение прерывания по завершению передачи
Запись в данный бит лог. 1 разрешает прерывание по флагу TXC. Прерывание по завершению передачи УСАПП генерируется, если TXCIE=1, флаг общего разрешения прерываний I=1 (в регистре SREG), а также установлен бит TXC в регистре UCSRA.

• **Bit 5 – UDRIE:** Разрешение прерывания по освобождению регистра данных УСАПП
Установка данного флага разрешает прерывание по флагу UDRE. Прерывание по освобождению регистра данных генерируется, если бит UDRIE=1, флаг общего разрешения прерываний I=1 (в регистре SREG) и установлен бит UDRE в регистре UCSRA.

• **Bit 4 – RXEN:** Разрешение работы приемника
Запись в данный бит лог. 1 приводит к разрешению работы приемника УСАПП. При этом, приемник формирует отключающий сигнал, который разрешает альтернативную функцию вывода RxD. Отключение приемника приводит к сбросу приемного буфера, теряя при этом значения флагов FE, DOR и UPE.

• **Bit 3 – TXEN:** Разрешение работы передатчика
Запись в данный бит лог. 1 разрешает работу передатчика УСАПП. После этого передатчик генерирует отключающий сигнал, который активизирует альтернативную функцию вывода TxD. Отключение передатчика (запись лог. 0 в TXEN) вступит в силу только по завершении генерации посылки, т.е. когда освободятся и сдвиговый регистр и буфер передатчика. После отключения вывод TxD возвращается к выполнению функции обычной линии ввода-вывода.

• **Bit 2 – UCSZ2:** Формат данных
Бит UCSZ2 вместе с битами UCSZ1:0 в регистре UCSRC задают количество бит данных в посылке, как для приемника, так и для передатчика.

• **Bit 1 – RXB8:** Значение 8-ого разряда принятых данных
RXB8 содержит значение 9-го бита принятой посылки с 9-битным форматом. Данный бит необходимо считать прежде, чем будут считаны младшие 8 бит из регистра UDR.

• **Bit 0 – TXB8:** 8-ой разряд передаваемых данных

TXB8 содержит значение 9-ого бита данных для передачи посылки с 9-битным форматом. Данный бит необходимо записать перед тем, как будут записаны младшие разряды данных в UDR.

Регистр С управления и статуса УСАПП – UCSRC

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

Регистр UCSRC совместно использует то же самое расположение ввода-вывода как Регистр UBRRH. См. “Доступ Регистры UBRRH/UCSRC” на разделе страницы 152, который описывает, как получить доступ к этому регистру.

• **Bit 7 – URSEL:** Выбор регистра
Этот бит выбирает между доступом к UCSRC или регистру UBRRH. URSEL должен быть 1 при записи UCSRC.

• **Bit 6 – UMSEL:** Выбор режима УСАПП
Данный бит позволяет переключаться между синхронным и асинхронными режимами последовательной связи.
Таблица 55. Установки бита UMSEL

UMSEL	Режим связи
0	Асинхронный
1	Синхронный

• **Bit 5:4 – UPM1:0:** Режим паритета
Данные бита разрешают и устанавливают тип генерируемого и контролируемого паритета. После разрешения паритета передатчик автоматически генерирует и передает бит паритета в каждой посылке. Приемник генерирует бит паритета для принятых данных и сравнивает его со значением принятого в этой посылке бита паритета, а по результату сравнения устанавливает флаг ошибки паритета UPE в регистре UCSRA.
Таблица 56. Установки бит UPM

UPM1	UPM0	Режим четности
0	0	Выключен
0	1	Зарезервирован
1	0	Включен, четная
1	1	Включен, нечетная

• **Bit 3 – USBS:** Выбор числа стоп-бит
Данный бит определяет сколько стоповых бит вставляет передатчик при генерации посылки. Приемник игнорирует эту настройку.
Таблица 57. Установки бита USBS

USBS	Число стоп-бит
0	1 бит
1	2 бита

• **Bit 2:1 – UCSZ1:0:** Формат данных
Биты UCSZ1:0 вместе с UCSZ2 в регистре UCSRB задают количество бит данных в посылке, как для приемника, так и для передатчика.
Таблица 58. Установки бит UCSZ

UCSZ2	UCSZ1	UCSZ0	Биты данных
0	0	0	5
0	0	1	6
0	1	0	7
0	1	1	8
1	0	0	зарезервировано
1	0	1	зарезервировано
1	1	0	зарезервировано
1	1	1	9

• **Bit 0 – UCPOL:** Полярность синхронизации

Данный бит используется только в синхронном режиме. Если используется асинхронный режим, то в данный бит необходимо записать лог. 0. В синхронном режиме бит UCPOL определяет соотношение между выборкой входящих данных и обновлением передаваемых данных и сигналом тактирования синхронной связи (XCK).

Таблица 59. Установки бит UCPOL

UCPOL	Изменение передаваемых данных на выходе TxD	Выборка принимаемых данных на входе RxD
0	Нарастающий фронт XCK	Падающий фронт XCK
1	Падающий фронт XCK	Нарастающий фронт XCK

Регистры скорости связи УСАПП - UBRRL и UBRRH

Bit	15	14	13	12	11	10	9	8	
	URSEL	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Регистр UBRRH совместно использует то же самое расположение ввода-вывода как Регистр UCSRC. См. “Доступ Регистры UBRRH/UCSRC” на разделе страницы 152, который описывает, как получить доступ к этому регистру.

• **Bit 15 – URSEL:** Выбор регистра
Этот бит выбирает между доступом к UBRRH или Регистру UCSRC.
URSEL должен быть нулем при записи UBRRH.

• **Bit 14:12 –** Зарезервированные разряды
Данные разряды зарезервированы для будущего использования. Для совместимости с последующими разработками необходимо записать лог. 0 в эти разряды во время записи в регистр UBRRH.

• **Bit 11:0 – UBRR11:0:** Регистр скорости связи УСАПП
UBRR - 12-разр. регистр, который задает значение скорости связи УСАПП. Регистр UBRRH содержит 4 старших разряда, а UBRRL 8 младших разрядов значения скорости УСАПП. Если во время передачи или приема изменить скорость связи, то сеанс связи будет нарушен. Запись в регистр UBRRL инициирует обновление предделителя скорости связи.

Примеры установок скоростей связи

В таблице 60 приведены примеры установок UBRR для генерации стандартных скоростей связи при типичных тактовых частотах микроконтроллера. Значения UBRR, которые дают результирующую скорость связи, отличающуюся не более чем на 0.5% от искомого значения, в таблице выделены жирным шрифтом. Более высокие погрешности также приемлемы, но приемник будет обладать меньшей помехоустойчивостью, особенно при передаче длинных посылок (см. "**Рабочий диапазон асинхронной связи**"). Значения погрешностей вычислены по следующему выражению:

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \bullet 100\%$$

Таблица 60. Примеры установок UBRR для типичных частот тактового генератора

Скорость, бит/сек	f_{osc} = 1.0000 MHz				f_{osc} = 1.8432 MHz				f_{osc} = 2.0000 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
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	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	–	–	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	–	–	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	–	–	–	–	–	–	0	0.0%	–	–	–	–
250k	–	–	–	–	–	–	–	–	–	–	0	0.0%
Max (1)	62.5 Kbps		125 Kbps		115.2 Kbps		230.4 Kbps		125 Kbps		250 Kbps	

1. UBRR = 0, Погрешность = 0.0%

Таблица 61. Примеры установок UBRR для типичных частот тактового генератора

Скорость, бит/сек	f_{osc} = 3.6864 MHz				f_{osc} = 4.0000 MHz				f_{osc} = 7.3728 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	–	–	0	-7.8%	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M	–	–	–	–	–	–	–	–	–	–	0	-7.8%
Max (1)	230.4 Kbps		460.8 Kbps		250 Kbps		0.5 Mbps		460.8 Kbps		921.6 Kbps	

1. UBRR = 0, Погрешность = 0.0%

Таблица 62. Примеры установок UBRR для типичных частот тактового генератора

Скорость, бит/сек	f_{osc} = 8.0000 MHz				f_{osc} = 11.0592 MHz				f_{osc} = 14.7456 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max (1)	0.5 Mbps		1 Mbps		691.2 Kbps		1.3824 Mbps		921.6 Kbps		1.8432 Mbps	

1. UBRR = 0, Погрешность = 0.0%

Таблица 63. Примеры установок UBRR для типичных частот тактового генератора

Скорость, бит/сек	f_{osc} = 16.0000 MHz				f_{osc} = 18.4320 MHz				f_{osc} = 20.0000 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max (1)	1 Mbps		2 Mbps		1.152 Mbps		2.304 Mbps		1.25 Mbps		2.5 Mbps	

1. UBRR = 0, Погрешность = 0.0%

Двухпроводный последовательный интерфейс TWI

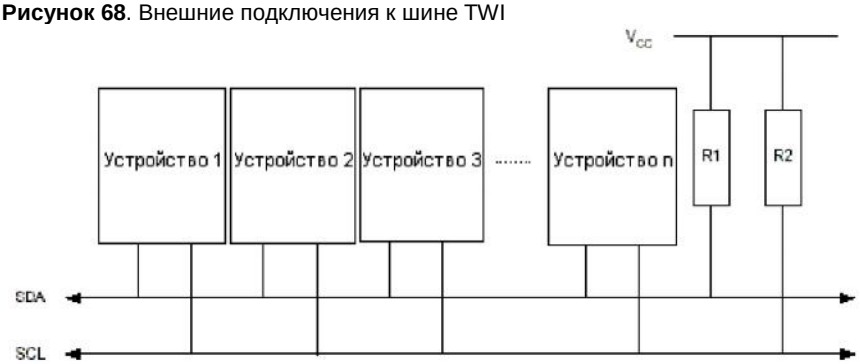
Отличительные особенности:

- **Гибкий, простой, при этом эффективный последовательный коммуникационный интерфейс, требующий только две линии связи**
- **Поддержка как ведущей, так и подчиненной работы**
- **Возможность работы, как приемника, так и как передатчика**
- **7-разр. адресное пространство позволяет подключить к шине до 128 подчиненных устройств**
- **Поддержка многомастерного арбитрирования**
- **Скорость передачи данных до 400 кГц**
- **Выходы драйверов с ограниченной скоростью изменения сигналов**
- **Схема шумоподавления повышает стойкость к выбросам на линиях шины**
- **Программируемый адрес для подчиненного режима с поддержкой общего вызова**
- **Пробуждение микроконтроллера из режима сна при определении заданного адреса на шине**

Определение шины TWI

Двухпроводной последовательный интерфейс TWI идеально подходит для типичных применений микроконтроллера. Протокол TWI позволяет проектировщику системы внешне связать до 128 различных устройств через одну двухпроводную двунаправленную шину, где одна линия - линия синхронизации SCL и одна - линия данных SDA. В качестве внешних аппаратных компонентов, которые требуются для реализации шины, необходимы только подтягивающий к плюсу питания резистор на каждой линии шины. Все устройства, которые подключены к шине, имеют свой индивидуальный адрес, а механизм определения содержимого шины поддерживается протоколом TWI.

Рисунок 68. Внешние подключения к шине TWI



Терминология TWI

Ниже приведены часто используемые в данном разделе термины.

Таблица 64. Терминология TWI

Термин	Описание
Ведущий	Устройство, которое инициирует и прекращает сеанс связи. На стороне ведущего также генерируется сигнал синхронизации SCL
Подчиненный	Устройство, которое адресуется ведущим устройством
Передатчик	Устройство, размещающее данные на шине
Приемник	Устройство, считывающее данные с шины

Внешнее электрическое соединение

Как показано на рисунке 86, обе линии шины подключены к положительной шине питания через подтягивающие резисторы. Среди всех совместимых с TWI устройствами в качестве драйверов шины используются транзистор или с открытым стоком или с открытым коллектором. Этим реализована функция монтажного И, которая очень важна для двунаправленной работы интерфейса. Низкий логический уровень на линии шины TWI генерируется, если одно или более из TWI-устройств выводит лог. 0. Высокий уровень на линии присутствует, если все TWI-устройства перешли в третье высокоимпедансное состояние, позволяя подтягивающим резисторам задать уровень лог. 1. Обратите внимание, что при подключении к шине TWI нескольких AVR- микроконтроллеров, для работы шины должны быть запитаны все из этих микроконтроллеров.

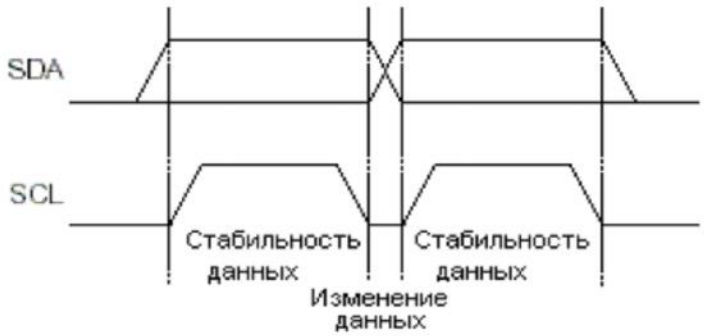
Количество устройств, которое может быть подключено к одной шине ограничивается предельно допустимой емкостью шины (400 пФ) и 7-разр. адресным пространством. Детальное описание электрических характеристик TWI приведено в параграфе "Характеристики двухпроводного последовательного интерфейса". Поддерживаются два различных набора технических требований, где один набор для шин со скоростью передачи данных ниже 100 кГц и один действителен для скоростей свыше 400 кГц.

Формат посылки и передаваемых данных

Передаваемые биты

Каждый передаваемый бит данных по шине TWI сопровождается импульсом на линии синхронизации. Уровень данных должен быть стабильным, когда на линии синхронизации присутствует лог. 1. Исключением для этого правила является генерация условий старта и останова сеанса связи.

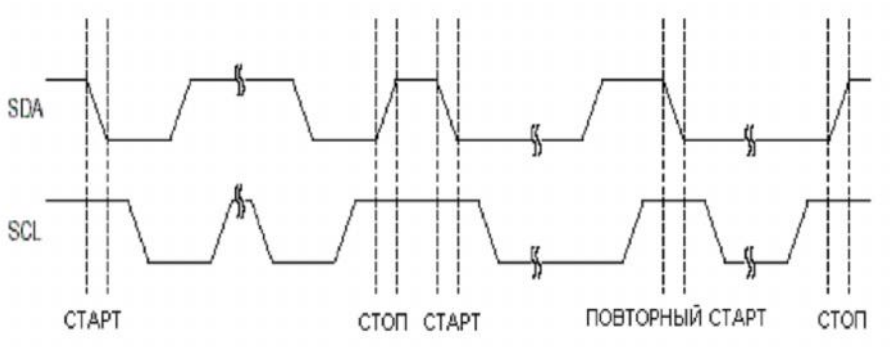
Рисунок 69. Действительность данных



Условия СТАРТа и ОСТАНОВа

Ведущее устройство инициирует и заканчивает передачу данных. Передача инициируется, когда ведущий формирует условие СТАРТа на шине, и прекращается, когда ведущий формирует на шине условие ОСТАНОВа. Между условиями СТАРТа и ОСТАНОВа шина считается занятой и в этом случае ни какой другой мастер не может осуществлять управляющие воздействия на шине. Существуют особые случаи, когда новое условие СТАРТа возникает между условиями СТАРТа и ОСТАНОВа. Данный случай именуется как условие "Повторного старта" и используется при необходимости инициировать мастером новый сеанс связи, не теряя при этом управление шиной. После "Повторного старта" шина считается занятой до следующего ОСТАНОВа. Это идентично поведению после СТАРТа, следовательно, при описании ссылка на условие СТАРТа распространяется и на "Повторный старт", если, конечно же, нет специального примечания. Как показано ниже, условия СТАРТа и ОСТАНОВа являются изменение логического уровня на линии SDA, когда на линии SCL присутствует лог. 1.

Рисунок 70. Условия СТАРТа, ПОВТОРНОГО СТАРТА и ОСТАНОВа



Формат адресного пакета

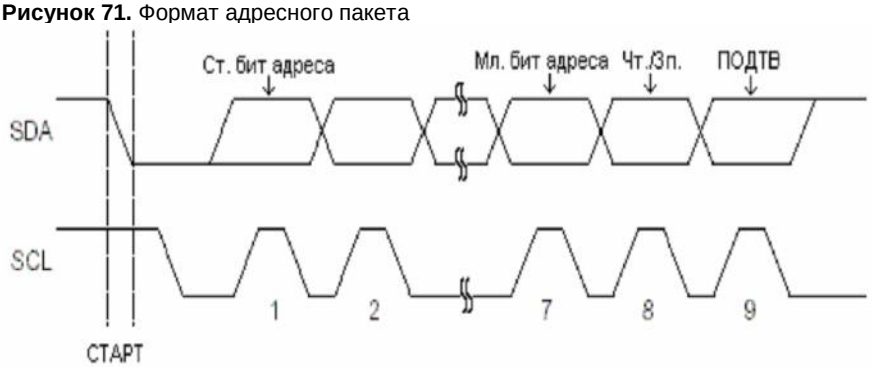
Все передаваемые адресные пакеты по шине TWI состоят из 9 бит, в т.ч. 7 бит адреса, один бит управления для задания типа операции ЧТЕНИЕ/ЗАПИСЬ и один бит подтверждения. Если бит ЧТЕНИЕ/ЗАПИСЬ = 1, то будет выполнена операция чтения, иначе - запись. Если подчиненный распознает, что к нему происходит адресация, то он должен сформировать низкий уровень на линии SDA на 9-ом цикле SCL (формирование бита подтверждения). Если адресуемое подчиненное устройство занято или по каким-либо другим причинам не может обслужить ведущее устройство, то на линии SDA необходимо оставить высокий уровень во время цикла подтверждения. Ведущий после этого может передать условие ОСТАНОВа или "Повторного старта" для инициации новой передачи. Адресный пакет, состоящий из адреса подчиненного устройства и бита ЧТЕНИЕ или ЗАПИСЬ, обозначим как ПОДЧИН_АДР+ЧТЕНИЕ или ПОДЧИН_АДР+ЗАПИСЬ, соответственно.

Старший разряд адресного байта передается первым. Нет никаких ограничений на выбор адреса подчиненного устройства, за исключением адреса 0000 000, который зарезервирован для общего вызова.

При определении общего вызова все подчиненные устройства должны ответить низким уровнем на линии SDA во время цикла подтверждения (ACK). Общий вызов необходимо использовать, если одно и тоже сообщение необходимо передать от ведущего к нескольким подчиненным устройствам. Если вслед за битом ЗАПИСИ передан адрес общего вызова, то все подчиненные устройства устанавливают низкий уровень на линии SDA для подтверждения общего вызова во время цикла подтверждения. Следующие пакеты данных будут приниматься всеми подчиненными устройствами, которые подтвердили общий вызов. Обратите внимание, что передача адреса общего вызова вслед за битом ЧТЕНИЕ бессмысленна, т.к. одновременное чтение нескольких подчиненных устройств одним ведущим не возможно.

Все адреса с форматом 1111 xx необходимо зарезервировать для будущего использования.

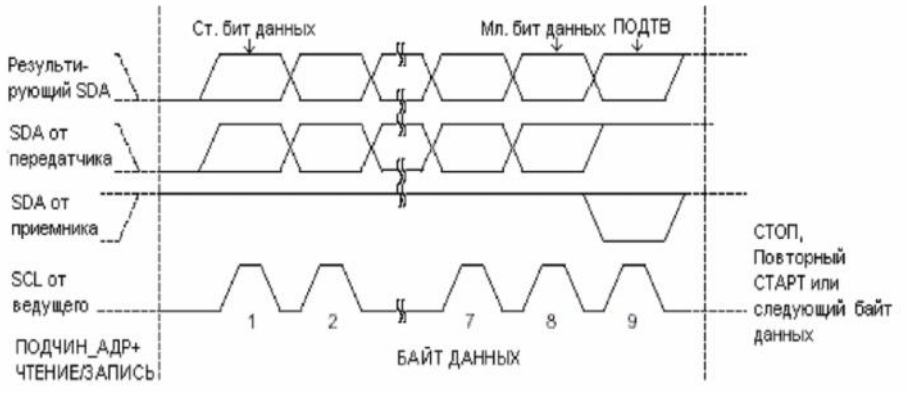
Рисунок 71. Формат адресного пакета



Формат пакета данных

Все пакеты данных, передаваемые по шине TWI, состоят из 9 бит, в т.ч. 1 байт данных и бит подтверждения. Во время передачи данных ведущее устройство генерирует синхронизацию, а также условия СТАРТа и ОСТАНОВа, при этом на приемник возлагается подтверждение приема. Подтверждение (ПОДТВ) сигнализируется приемником выводом низкого уровня на линию SDA во время 9-го такта сигнала SCL. Если приемник оставляет линию SDA в высоком состоянии, то этот сигнализирует о том, что подтверждения не было (НЕТ ПОДТВ). После получения приемником последнего байта или если по каким-либо причинам не имеется возможности далее принимать данные он должен информировать передатчика отправкой бита НЕТ ПОДТВ (нет подтверждения) после последнего байта. Старший бит данных передается первым.

Рисунок 72. Формат пакета данных

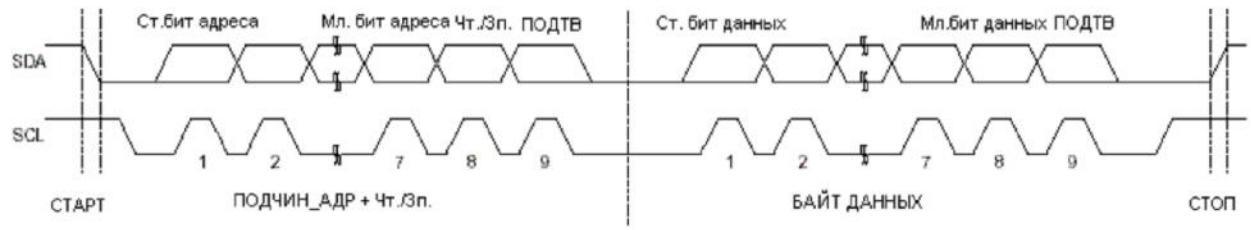


Сочетание пакетов адреса и данных во время сеанса связи

Сеанс связи обычно состоит из условия СТАРТа, ПОДЧИН_АДР+ЧТЕНИЕ/ЗАПИСЬ, одного или более пакетов данных и условия ОСТАНОВа. Передача пустого сообщения, которое состоит из условия СТАРТа, переданного вслед за условием ОСТАНОВа, является недопустимым. Обратите внимание, что монтажное "И" на линии SCL может использоваться для реализации подтверждения связи между ведущим и подчиненным. Подчиненный может продлить низкое состояние на линии SCL путем установки лог. 0 на выводе SCL. Данный способ полезно использовать, если установленная скорость связи мастером является повышенной по отношению к подчиненному или если подчиненному требуется дополнительное время на обработку между приемами данных. Починенный, продлевающий низкое состояние на линии SCL, не будет оказывать влияние на длительность высокого состояния SCL, которая определяется мастером. Как следствие, подчиненный может снизить скорость передачи данных, продлевая рабочий цикл SCL.

На рисунке 73 показана типичная передача данных. Обратите внимание, что несколько байт данных могут быть переданы между условиями ПОДЧИН_АДР+ЧТЕНИЕ/ЗАПИСЬ и СТОП в зависимости от программного протокола, реализованного в прикладной программе.

Рисунок 73. Типичная передача данных



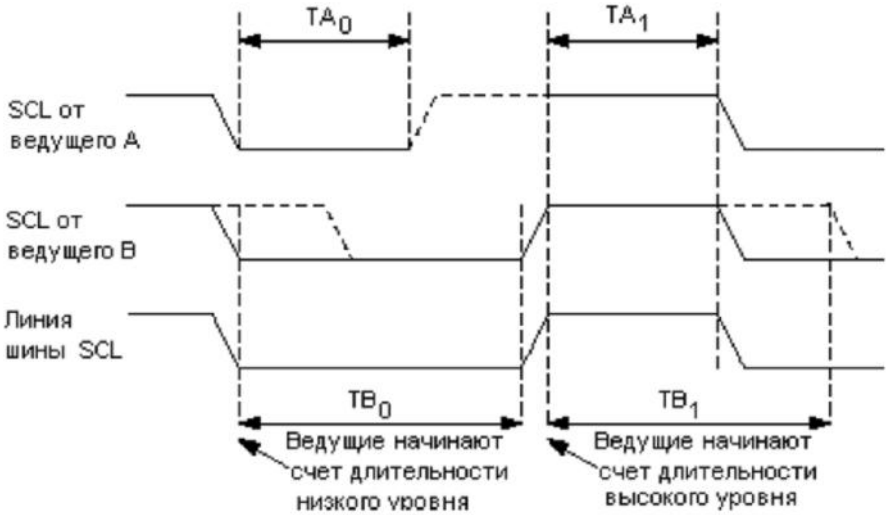
Системы многомастерных шин, арбитраж и синхронизация

Протокол TWI допускает размещение нескольких ведущих устройств на одной шине. Чтобы гарантировать нормальность процесса передачи, даже если два и более ведущих устройств инициируют передачу в одно и то же время, было предпринято несколько мер. Вот две проблемы, которые необходимо решить в многомастерных системах:

- Алгоритм необходимо реализовать так, чтобы только одно ведущее устройство могло завершить передачу. Все остальные ведущие устройства должны прекратить передачу после обнаружения потери процесса выбора. Данный процесс выбора носит название арбитрирование. Если ведущее устройство обнаруживает потерю процесса арбитрирования, то он должен сразу перейти в подчиненный режим, чтобы проверить не к нему ли обращается ведущее устройство, которое выиграло процесс арбитрирования. Факт начала одновременной передачи несколькими ведущими устройствами не должен обнаружиться подчиненными, т.е., передаваемые данные должны быть повреждены.
- Разные ведущие устройства могут использовать разные частоты сигнала SCL. Необходимо придумать схему, которая позволяла бы засинхронизировать тактовые сигналы всех ведущих устройств.

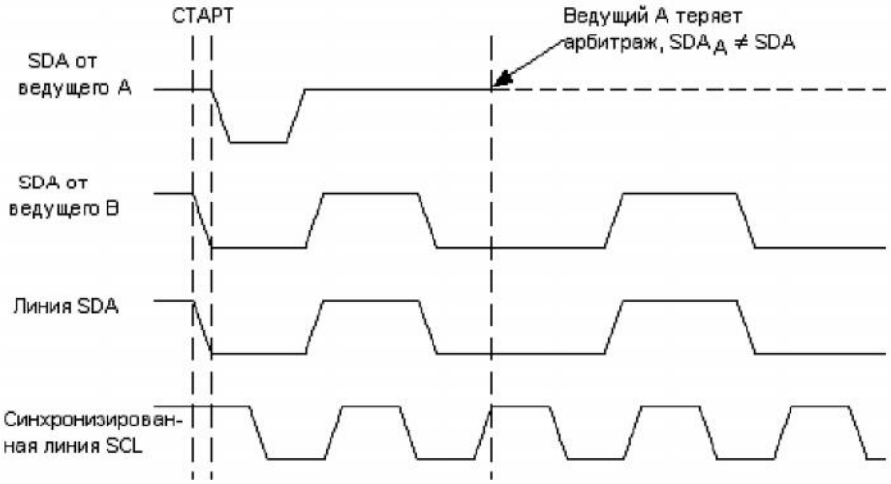
Использование принципа монтажного "И" позволяет решить обе эти проблемы. Соединение тактовых сигналов всех ведущих устройств по схеме монтажного "И" означает, что длительность результирующего единичного импульса будет равна длительности самого короткого единичного импульса в этом соединении. Длительность низкого уровня результирующего тактового сигнала равна длительности низкого уровня тактового сигнала того ведущего устройства, у которого она максимальная. Обратите внимание, что все ведущие устройства, подключенные к линии SCL, начинают счет времени окончания периодов с высоким и низким состояниями, когда результирующий сигнал SCL переходит в высокое или низкое состояние, соответственно.

Рисунок 74. Синхронизация на линии SCL между несколькими ведущими устройствами



Арбитрация реализована путем контроля состояния линии SDA всеми ведущими, выводящих данные. Если уровень присутствующий на линии SDA не совпадает со значением, который передал ведущий, то данный ведущий теряет арбитрацию. Обратите внимание, что арбитрация теряется только в том случае, если ведущий передал лог. 1, а фактически на линии SDA присутствовал лог. 0. После потери ведущим арбитрации он незамедлительно переходит в подчиненный режим для определения не к нему ли адресуется выигравший арбитрацию ведущий? Ведущие, которые проиграли процесс арбитрации, могут продолжать генерировать тактовый сигнал до окончания передачи текущего пакета адреса или данных, но при этом они должны выводить высокий уровень на линию SDA. Арбитрация выполняется то тех пор, пока останется активным только один ведущий и для этого в ряде случаев может быть передано много бит. Если несколько ведущих пытаются адресоваться к одному и тому же подчиненному, то процесс арбитрации переносится на пакет данных.

Рисунок 75. Арбитрирование двух мастеров



Обратите внимание, что арбитрация не выполняется между передачей:

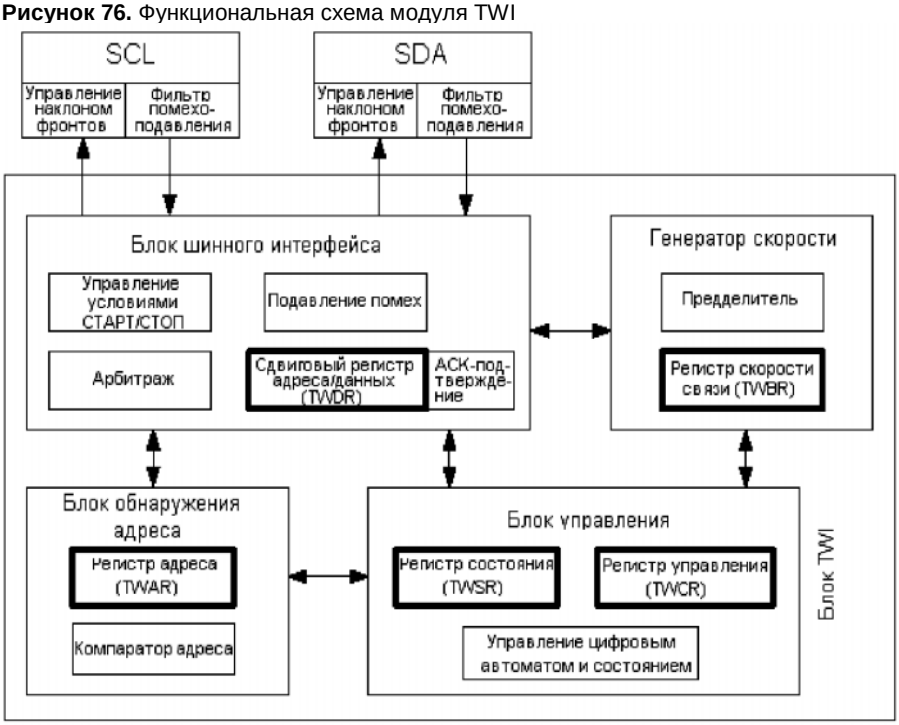
- Условия ПОВТОРНЫЙ СТАРТ и бита данных
- Условия СТОП и бита данных
- Условия ПОВТОРНЫЙ СТАРТ и СТОП

Гарантирование невозможности возникновения данных условий возлагается на программное обеспечение пользователя. Этим подразумевается, что во многомастерных системах должны использоваться одинаковое сочетание пакетов данных и ПОДЧИН_АДР+ЧТЕНИЕ/ЗАПИСЬ. Или иначе: любой сеанс связи должен состоять из одинакового числа пакетов данных, в противном случае результат арбитрации будет неопределенным.

Обзор модуля TWI

Модуль TWI состоит из нескольких подмодулей (см. рисунок 76). Все регистры выделенные жирной линией доступны через шину данных микроконтроллера.

Рисунок 76. Функциональная схема модуля TWI



Выводы SCL и SDA

Данные выводы связывают двухпроводной интерфейс микроконтроллера с остальными микроконтроллерами в системе. Драйверы выходов содержат ограничитель скорости изменения фронтов для выполнения требований к TWI. Входные каскады содержат блок подавления помех, задача которого состоит в игнорировании импульсов длительностью менее 50 нс. Обратите внимание, что к каждой из этих линий можно подключить внутренний подтягивающий резистор путем установки разрядов PORTD.0 (SCL), PORTD.1 (SDA) (см. также "**Порты ввода-вывода**"). Использование встроенных подтягивающих резисторов в ряде случаев позволяет отказаться от применения внешних.

Блок генератора скорости связи

Данный блок управляет периодом импульсов SCL в режиме ведущего устройства. Период SCL задается регистром скорости TWI (TWBR) и значением бит управления предделителем в регистре состояния TWI (TWSR). В подчиненном режиме значения скорости или установки предделителя не оказывают влияния на работу, но частота синхронизации ЦПУ подчиненного устройства должна быть минимум в 16 раз выше частоты SCL. Обратите внимание, что подчиненные могут продлевать длительность низкого уровня на линии SCL, тем самым уменьшая среднюю частоту синхронизации шины TWI. Частота SCL генерируется в соответствии со следующим выражением:

$$f_{SCL} = \frac{f_{ЦПУ}}{16 + 2 \cdot (TWBR) \cdot 4^{TWPS}},$$

Где
TWBR - значение регистра скорости TWI;
TWPS - значение бит предделителя в регистре состояния TWI.

Прим.: значения подтягивающего резистора должны быть выбраны согласно частоте SCL и емкостной загрузке линии шины. См. Таблицу 101 на странице 245 для значения подтягивающего резистора."

Блок шинного интерфейса

Данный блок содержит сдвиговый регистр адреса и данных (TWDR), контроллер СТАРТа/СТОПа и схему арбитрации. TWDR содержит передаваемый байт адреса или данных, или принятый байт адреса или данных. Помимо 8-разр. регистра TWDR в состав блока шинного интерфейса также входит регистр, хранящий значение передаваемого или принятого бита (НЕТ) ПОДТВ. К данному регистру нет прямого доступа со стороны программного обеспечения. Однако во время приема он может устанавливаться или сбрасываться путем манипуляций с регистром управления TWI (TWCR). В режиме передатчика значение принятого бита (НЕТ) ПОДТВ можно определить по значению регистра TWSR.

Контроллер СТАРТа/СТОПа отвечает за генерацию и детекцию условий СТАРТ, ПОВТОРНЫЙ СТАРТ и СТОП. Контроллер СТАРТа/СТОПа позволяет обнаружить условия СТАРТ и СТОП, даже если микроконтроллер находится в одном из режимов сна. Этим обеспечивается возможность пробуждения микроконтроллера по запросу ведущего шины.

Если TWI инициировал передачу в качестве ведущего, то схема арбитрации непрерывно контролирует передачу, определяя возможность дальнейшей передачи. Если TWI теряет арбитрацию, то блок формирует соответствующий сигнал блоку управления, который выполняет адекватные действия и генерирует соответствующий код состояния.

Блок обнаружения адреса

Блок обнаружения адреса проверяет равен ли принятый адрес значению 7-разр. адреса из регистра TWAR. Если установлен бит разрешения обнаружения общего вызова TWGCE в регистре TWAR, то все входящие адресные биты будут дополнительно сравниваться с адресом общего вызова. При адресном совпадении подается сигнал блоку управления, что позволяет выполнить ему необходимые действия. В зависимости от установки регистра TWCR подтверждение адреса TWI может происходит, а может и нет. Блок обнаружения адреса способен функционировать даже, когда микроконтроллер переведен в режим сна, тем самым позволяя возобновить нормальную работу микроконтроллера по запросу мастера шины. Если при адресном совпадении TWI в экономичном режиме микроконтроллера, т.е. когда инициируется возобновление работы микроконтроллера, возникает другое прерывание (например, INT0), то TWI прекращает работу и возвращается к состоянию холостого хода (Idle). Если возникновение данного эффекта нежелательно, то необходимо следить, чтобы во время обнаружения адресования, когда микроконтроллер находится в режиме выключения (Power-down), было разрешено только одно прерывание.

Блок управления

Блок управления наблюдает за шиной TWI и генерирует отклики в соответствии с установками регистра управления TWI (TWCR). Если на шине TWI возникает событие, которое требует внимания со стороны программы, то устанавливается флаг прерывания TWINT. Следующим тактом обновляется содержимое регистра статуса TWI - TWSR, в котором будет записан код, идентифицирующий возникшее событие. Даная информация хранится в TWSR только тогда, когда установлен флаг прерывания TWI.

Остальное время в регистре TWSR содержится специальный код состояния, который информирует о том, что нет информации о состоянии TWI. До тех пор пока установлен флаг TWINT линия SCL остается в низком состоянии. Этим обеспечивается возможность завершить программе все задачи перед продолжением сеанса связи.

Флаг TWINT устанавливается в следующих ситуациях:
- После передачи условия СТАРТ/ПОВТОРНЫЙ_СТАРТ
- После передачи ПОДЧИН_АДР+ЧТЕНИЕ/ЗАПИСЬ
- После передачи адресного байта
- После потери арбитрирования
- После того как TWI адресован собственным подчиненным адресом или общим вызовом
- После приема байта данных
- После приема условия СТОП или ПОВТОРНЫЙ_СТАРТ в режиме подчиненной адресации
- После возникновения ошибки по причине некорректного условия СТАРТ или СТОП

Описание регистров TWI

Регистр скорости связи шины TWI – TWBR

Bit	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• **Bits 7..0 –**Биты регистра скорости связи шины TWI
TWBR задает коэффициент деления частоты генератора скорости связи. Генератор частоты скорости связи - делитель частоты, который формирует сигнал синхронизации SCL в режимах "Ведущий". В разделе "Блок генератора скорости связи" показана методика вычисления скоростей связи. на странице 170

Регистр управления шиной TWI – TWCR

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Регистр TWCR предназначен для управления работой TWI. Он используется для разрешения работы TWI, для инициации сеанса связи ведущего путем генерации условия СТАРТ на шине, для генерации подтверждения приема, для генерации условия СТОП и для останова шины во время записи в регистр TWDR. Он также сигнализирует о попытке ошибочной записи в регистр TWDR, когда доступ к нему был запрещен.
• **Bit 7 – TWINT:** Флаг прерывания TWI
Данный бит устанавливается аппаратно, если TWI завершает текущее задание и ожидает реакции программы. Если бит I в SREG и бит TWIE в TWCR установлены, то микроконтроллер переходит на вектор прерывания TWI. Линия SCL остается в низком состоянии, пока установлен флаг TWINT. Флаг TWINT сбрасывается программно путем записи в него логической 1. Обратите внимание, что данный флаг сбрасывается не автоматически при переходе на вектор прерывания. Также нужно учесть, что очистка данного флага приводит к возобновлению работы TWI. Из этого следует, что программный сброс данного флага необходимо выполнить после завершения опроса регистров TWAR, TWSR и TWDR.

• **Bit 6 – TWEA:** Бит разрешения подтверждения

Бит TWEA управляет генерацией импульса подтверждения. Если в бит TWEA записана лог. 1, то импульс ПОДТВ генерируется на шине TWI, если выполняется одно из следующих условий:
1. Принят собственный подчиненный адрес.
2. Принят общий вызов, когда установлен бит TWGCE в регистре TWAR.
3. Принят байт данных в режиме ведущего приемника или подчиненного приемника.

Запись лог. 0 в бит TWEA позволяет временно отключиться от двухпроводной последовательной шины. Для возобновления распознавания адреса необходимо записать в данный бит лог.1.

• **Bit 5 – TWSTA:** Бит условия СТАРТ
Программист должен установить данный бит при необходимости стать ведущим на двухпроводной последовательной шине. TWI аппаратно проверяет доступность шины и генерирует условие СТАРТ, если шина свободна. Однако если шина занята, то TWI ожидает появления условия СТОП, а затем генерирует новое условие СТАРТ для перехвата состояния ведущего шины. TWSTA необходимо сбрасывает программно после передачи условия СТАРТ.

• **Bit 4 – TWSTO:** Бит условия СТОП
Установка бита TWSTO в режиме ведущего приводит к генерации условия СТОП на двухпроводной последовательной шине. Если на шине выполняется условие СТОП, то бит TWSTO сбрасывается автоматически. В подчиненном режиме установка бита TWSTO может использоваться для выхода из условия ошибки. В этом случае условие СТОП не генерируется, но интерфейс TWI возвращается к хорошо сконфигурированному безадресному подчиненному режиму и переводит линии SCL и SDA в высокоимпедансное состояние.

• **Bit 3 – TWWC:** Флаг ошибочной записи
Бит TWWC устанавливается при попытке записи в регистр данных TWDR, когда TWINT имеет низкий уровень. Флаг сбрасывается при записи регистра TWDR, когда TWINT = 1.

• **Bit 2 – TWEN:** Бит разрешения работы TWI
Бит TWEN разрешает работу TWI и активизирует интерфейс TWI. Если бит TWEN установлен, то TWI берет на себя функции управления линиями ввода-вывода SCL и SDA. При этом разрешается работа ограничителей скорости изменения фронтов и помехоподавляющих фильтров. Если данный бит равен нулю, то TWI отключается и все передачи прекращаются независимо от состояния работы.

• **Bit 1 – Res:** Резервный бит
Данный бит является резервным и считывается как 0.

• **Bit 0 – TWIE:** Разрешение прерывания TWI
Если в данный бит записана лог. 1 и установлен бит I в регистре SREG, то запрос на прерывание TWI будет генерироваться до тех пор, пока установлен флаг TWINT.

Регистр состояния TWI – TWSR

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

• **Bits 7..3 – TWS:** Состояние TWI
Данные 5 бит отражают состояние логики блока TWI и двухпроводной последовательной шины. Различия в кодах состояния будут представлены далее в этом разделе. Обратите внимание, что считываемое значение из регистра TWSR содержит и 5-разр. код состояния и 2-разр. значение, управляющее предделителем. Программист должен маскировать к 0 биты предделителя во время проверки бит состояния. В этом случае проверка состояния не будет зависеть от настройки предделителя.

• **Bit 2 – Res:** Резервный бит
Данный бит является резервным и считывается как 0.

• **Bits 1..0 – TWPS:** Биты предделителя TWI
Данные биты отличаются полным доступом (чтение/запись) и позволяют управлять предделителем скорости связи.
Таблица 65. Предделитель скорости связи TWI

TWPS1	TWPS0	Значение предделения
0	0	1
0	1	4
1	0	16
1	1	64

Формула для вычисления скорости связи представлена в разделе "Блок генератора скорости связи". Значение бит TWPS1..0 используется в ней

Регистр данных шины TWI – TWDR

Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

В режиме передатчика регистр TWDR содержит следующий байт для передачи. В режиме приемника регистр TWDR содержит последний принятый байт. Запись в регистр возможна только когда TWI не выполняет процесс сдвига данных. Такое состояние наступает, когда происходит аппаратная установка флага прерывания TWINT. Обратите внимание, что регистр данных не может инициализироваться пользователем перед возникновением первого прерывания. Данные в регистре TWDR остаются стабильными пока установлен бит TWINT. Во время сдвига последовательной передачи данных одновременно происходит сдвиг для последовательного ввода. TWDR всегда содержит последний байт представленный на шине, исключая ситуацию возобновления нормальной работы микроконтроллера по прерыванию TWI. В этом случае состояние TWDR является неопределенным. В случае потери арбитрирования шины данные, передаваемые от ведущего к подчиненному, не теряются. Управление битом ПОДТВ происходит автоматически под управлением схемы TWI, а ЦПУ непосредственного доступа к биту ПОДТВ не имеет.
• **Bits 7..0 – TWD:** Регистр данных шины TWI
Данные 8 бит составляют байт данных, который необходимо передать следующим, или последний принятый байт по двухпроводной последовательной шине.

Регистр подчиненного адреса шины TWI – TWAR

Bit	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

Если TWI настроен на режим подчиненного передатчика или приемника, то будет реагировать только на адрес, записанный в этот регистр (в 7 старших разрядов TWAR). В остальных режимах ведущего данный регистр не используется. В многомастерных системах регистр TWAR устанавливается в том ведущем, к которому адресуются как к подчиненному другие ведущие шины.

Мл. разряд регистра TWAR используется для разрешения обнаружения адреса общего вызова ($00). Специальный компаратор выполняет сравнение подчиненного адреса (или адреса общего вызова) с принятым адресом. Если обнаруживается совпадение, то генерируется запрос на прерывание.

• **Bits 7..1 – TWA:** Регистр подчиненного адреса TWI
Данные семь бит составляют подчиненный адрес блока TWI.

• **Bit 0 – TWGCE:** Бит разрешения обнаружения общего вызова по шине TWI
После установки данного бита разрешается работа схемы обнаружения общего вызова, передаваемого по двухпроводной последовательной шине.

Рекомендации по использованию TWI

TWI ориентирован на передачу данных в байтном формате с управлением по прерываниям. Прерывания возникают после обнаружения одного из событий на шине, например, прием байта или передача условия СТАРТ. Управление TWI по прерываниям позволяет освободить программное обеспечение на выполнение других задач во время передачи байта данных. Обратите внимание, что установка флага TWINT приводит к генерации запроса на прерывание только в том случае, когда установлен бит разрешения прерывания TWIE в регистре TWCR, а также разрешена работа прерываний установкой бита в регистре SREG. Если бит TWIE сброшен, то состояние TWINT должно отслеживаться программно для оценки ситуации на шине TWI.

После установки флага TWINT интерфейс TWI приостанавливает работу и ожидает реакции программы. В этом случае регистр статуса TWI (TWSR) содержит значение, которое индицирует текущее состояние шины TWI. Исходя из этого программа задает дальнейшее поведение шины TWI, манипулируя регистрами TWCR и TWDR.

На рисунке 77 показан простой пример подключения к шине TWI. Здесь предполагается, что мастер желает передать один байт данных подчиненному. Данное описание весьма общее, а более подробно объяснение приводится далее в этом разделе.

Рисунок 77. Последовательность обслуживания TWI при типичной передаче

1. Первым шагом работы TWI является передача условия СТАРТ. Это инициируется путем записи специфического значения в TWCR. О значении, которое необходимо записать будет сказано позже. Однако, необходимо следить, чтобы в записываемом в регистр значении был установлен бит TWINT. Запись лог. 1 в TWINT сбрасывает этот флаг. TWI не начнет работу до тех пор пока будет установлен флаг TWINT в регистре TWCR. Сразу после сброса TWINT начинается передача условия СТАРТ.
2. После передачи условия СТАРТ устанавливается флаг TWINT в регистре TWCR, а содержимое TWSR обновляется значением кода состояния, индицирующего об успешной передачи условия СТАРТ.
3. В программе необходимо выполнить проверку значения TWSR, чтобы убедится в том, что условие СТАРТ было успешно передано. Если TWSR индицирует прочую ситуацию, то программа выполняет особые действия, например, вызывает процедуру обработки ошибочных ситуаций. Если код состояния имеет ожидаемое значение, то выполняется загрузка условия ПОДЧИН_АДР + ЗАПИСЬ в TWDR. Необходимо помнить, что TWDR используется для хранения как адреса, так и данных. После загрузки в TWDR желаемого значения ПОДЧИН_АДР + ЗАПИСЬ в регистр TWCR должно быть записано специфическое значение, которое служит командой для передачи значения ПОДЧИН_АДР + ЗАПИСЬ, хранящегося в TWDR. Какое именно значение необходимо записать будет сказано позже. Однако необходимо следить, чтобы в записываемом в регистр значении был установлен бит TWINT. Запись лог. 1 в TWINT приводит к сбросу этого флага. TWI не начнет работу до тех пор пока установлен бит TWINT в регистре TWCR. Сразу после сброса флага TWINT инициируется передача адресного пакета.
4. После передачи адресного пакета устанавливается флаг TWINT в регистре TWCR, а содержимое регистра TWSR обновляется кодом состояния, индицирующего успешность передачи адресного пакета. В коде состояния также отражается было ли подтверждение приема адресного пакета со стороны подчиненного или нет.
5. Выполняется программная проверка значения TWSR, чтобы убедиться в успешности передачи адресного пакета и что бит подтверждения ПОДТВ имеет ожидаемое значение. Если TWSR индицирует иную ситуацию, то при необходимости выполняются особые действия, например, вызывается процедура обработки ошибочных ситуаций. Если же код состояния имеет ожидаемое значение, то программа записывает пакет данных в TWDR. Впоследствии в регистр TWCR записывается специфическое значение, которое служит командой для TWI и вызывает аппаратную передачу данных, записанных в TWDR. Какое именно значение необходимо записать, будет сказано позже. Однако необходимо учесть, что в записываемом значении должен быть установлен бит TWINT. Запись лог. 1 в TWINT приводит к сбросу этого флага. TWI не начнет работу до тех пор пока будет установлен бит TWINT в регистре TWCR. Сразу после сброса TWINT начинается передача пакета данных.

6. После передачи пакета данных устанавливается флаг TWINT в регистре TWCR, а содержимое регистра TWSR обновляется значением кода состояния, который сигнализирует об успешной передачи пакета данных. В коде состояния также отражается было ли принято подтверждение от подчиненного или нет.
7. Выполняется программная проверка значения в TWSR, чтобы убедиться в успешности передачи пакета данных и в том, что бит ПОДТВ имеет ожидаемое значение. Если TWSR индицирует иную ситуацию, то программа выполняет особые действия, в т.ч. вызывает процедуру обработки прерывания. Если код состояния имеет ожидаемое значение, то выполняется запись специального значения в TWCR, которое служит командой для TWI и инициирует передачу условия СТОП. Какое именно значение необходимо записать, сказано далее. Однако следует учесть, что во время записи должна быть произведена установка бита TWINT. Запись лог. 1 в TWINT приводит к очистке этого флага. TWI не начнет работу до тех пор, пока установлен бит TWINT в регистре TWCR. Сразу после сброса флага TWINT инициируется передача условия СТОП. Обратите внимание, что флаг TWINT НЕ устанавливается по завершении передачи условия СТОП.

Не смотря на простоту изложенного примера, он показывает принципы, положенные в основу любой передачи через TWI. Из вышеизложенного можно сделать следующие выводы:

• По завершении работы TWI устанавливается флаг TWINT и далее ожидается реакция со стороны программы. Линия находится в низком состоянии, пока сброшен флаг TWINT.

• Если флаг TWINT установлен, то пользователь может обновлять любой из регистров TWI значением, которое относится к следующему этапу работы шины TWI. Например, в TWDR загружается значение, которое необходимо передать на следующем цикле шины.

• После обновления всех регистров TWI и завершении других задач выполняется запись в TWCR. Во время записи TWCR необходимо, чтобы был установлен бит TWINT. В этом случае запись лог. 1 в TWINT приведет к сбросу данного флага. TWI выполняет действия в соответствии установкой регистра TWCR.

Далее показан пример на Ассемблере и Си. В примере предполагается, что все символьные обозначения определены в присоединенном файле.

№	Пример кода на Ассемблере	Пример кода на Си	Комментарий
1	**ldi** r16, (1<<TWINT)|(1<<TWSTA)| (1<<TWEN) **out** TWCR, r16	TWCR = (1<<TWINT)|(1<<TWSTA)| (1<<TWEN)	Передача условия СТАРТ
2	wait1: **in** r16,TWCR 2 **sbrs** r16,TWINT 2 **rjmp** wait1	**while** (!(TWCR & (1<<TWINT))) ;	Ожидание установки флага TWINT. Этим индицируется завершение передачи условия СТАРТ
3	**in** r16,TWSR **andi** r16, 0xF8 **cpi** r16, START **brne** ERROR	**if** ((TWSR & 0xF8) != START) ERROR();	Проверка кода состояния TWI. Маскир. бит предделителяЕсли код состояния не равен СТАРТ, то переход на ERROR 4
	ldi r16, SLA_W **out** TWDR, r16 **ldi** r16, (1<<TWINT) | (1<<TWEN) **out** TWCR, r16	TWDR = SLA_W; TWCR = (1<<TWINT) | (1<<TWEN);	Загрузка ПОДЧИН_АДР + ЗАПИСЬ в регистр TWDR. Сброс бита TWINT в TWCR для начала передачи адреса
4	wait2: **in** r16,TWCR **sbrs** r16,TWINT **rjmp** wait2	**while** (!(TWCR & (1<<TWINT))) ;	Ожидание установки флага TWINT. Этим сигнализируется завершение передачи ПОДЧИН_АДР + ЗАПИСЬ и получение/неполучение подтверждения (ПОДТВ/НЕТ ПОДТВ).
5	**in** r16,TWSR **andi** r16, 0xF8 **cpi** r16, MT_SLA_ACK **brne** ERROR	**if** ((TWSR & 0xF8) != MT_SLA_ACK) ERROR();	Проверка значения регистра состояния.Маскирование бит предделителяЕсли состояние отличается от MT_SLA_ACK, то переход на ERROR
	ldi r16, DATA **out** TWDR, r16 **ldi** r16, (1<<TWINT) | (1<<TWEN) **out** TWCR, r16	TWDR = DATA; TWCR = (1<<TWINT) | (1<<TWEN);	Загрузка данных в TWDRСброс флага TWINT в TWCR для начала передачи данных
6	wait3: **in** r16,TWCR **sbrs** r16,TWINT **rjmp** wait3	**while** (!(TWCR & (1<<TWINT))) ;	Ожидание установки флага TWINTЭтим индицируется, что данные были переданы и принято/не принято подтверждение (ПОТДВ/НЕТ ПОДТВ).
7	**in** r16,TWSR **andi** r16, 0xF8 **cpi** r16, MT_DATA_ACK **brne** ERROR	**if** ((TWSR & 0xF8) != MT_DATA_ACK) ERROR();	Проверка значения регистра состояния TWI. Маскирование бит предделителя. Если состояние отличается от MT_DATA_ACK, то переход на ERROR
	ldi r16, (1<<TWINT)|(1<<TWEN)| (1<<TWSTO) **out** TWCR, r16	TWCR = (1<<TWINT)|(1<<TWEN)| (1<<TWSTO);	Передача условия СТОП

Режимы передачи

TWI может работать в одном из 4-х режимов работы. Они называются: ведущий передатчик (MT), ведущий приемник (MR), подчиненный передатчик (ST) и подчиненный приемник (SR). Некоторые из этих режимов могут использоваться в рамках одного и того же приложения. Например, TWI может использовать режим MT для записи данных в 2-пров. Последовательное EEPROM, а режим MR для считывания данных из EEPROM . Если в системе имеются другие ведущие (мастера), один из которых передает данные, то у остальных используется режим SR. Какой из режимов должен использоваться определяется программно.

Далее описаны каждый из этих режимов. Возможные значения кодов состояния представлены рядом с рисунками, детализирующих процесс передачи данных в каждом из режимов. На рисунках используются следующие аббревиатуры:
СТАРТ: Условие СТАРТ
ПОВТ СТАРТ: Условие Повторный СТАРТ
ЧТЕНИЕ: Бит "Чтение" (высокий уровень на SDA)
ЗАПИСЬ: Бит "Запись" (низкий уровень на SDA)
ПОДТВ: Бит подтверждения (низкий уровень на SDA)
НЕТ ПОДТВ: Нет бита подтверждения (высокий уровень на SDA)
ДАННЫЕ: 8-разр. байт данных
СТОП: Условие СТОП
ПОДЧИН_АДР: Подчиненный адрес

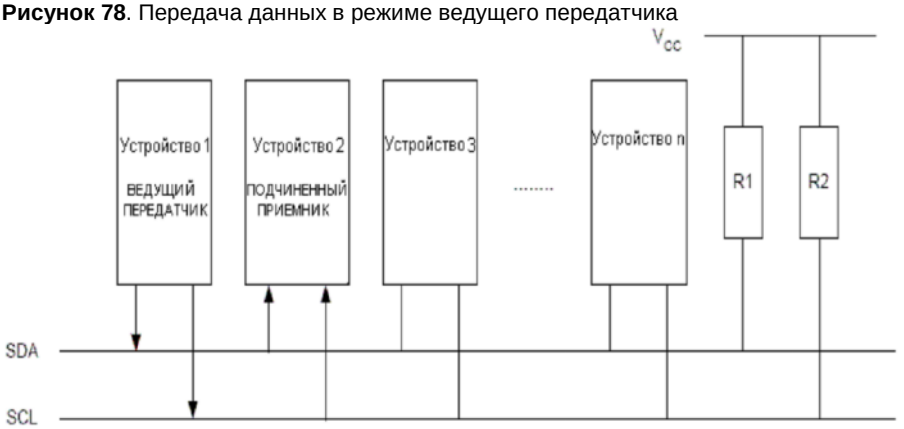
На рисунках 79 - 85 окружности используются для индикации установки флага TWINT. Число, записанное внутри окружности, является кодом состояния из регистра TWSR, в котором замаскированы к нулю биты предделителя. В данном состоянии ожидается действие со стороны программы для завершения передачи TWI. Передача TWI приостанавливается до тех пор, пока программно не будет сброшен флаг TWINT.

После установки флага TWINT по значению кода состояния из регистра TWSR определяется, какое действие выполнить программе. В таблицах 88-91 представлена информация о том, какие программные действия должны быть предприняты при различных значениях кода состояния. Обратите внимание, что в таблице биты предделителя замаскированы нулевыми значениями.

Режим ведущего передатчика

В режиме ведущего передатчика байты данных передаются подчиненному приемнику (см. рисунок 78). Для ввода режима ведущего необходимо передать условие СТАРТ. Формат следующего адресного пакета определяет какой режим вводится: ведущий передатчик или ведущий приемник. Если передается ПОДЧИН_АДР+ ЗАПИСЬ, то вводится режим MT (ведущий передатчик), а если ПОДЧИН_АДР + ЧТЕНИЕ, то вводится режим MR (ведущий приемник). Все упоминаемые в этом разделе коды состояния в позиции бит предделителя имеют нулевые значения.

Рисунок 78. Передача данных в режиме ведущего передатчика



Передача условия СТАРТ инициируется путем записи в TWCR следующего значения:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	1	0	x	1	0	x

Для разрешения работы двухпроводного последовательного интерфейса необходимо установить бит TWEN. Запись лог. 1 в TWSTA инициирует передачу условия СТАРТ, а запись лог. 1 в TWINT приводит к сбросу флага TWINT. После записи данного значения TWI тестирует двухпроводную последовательную шину и генерирует условие СТАРТ сразу после освобождения шины. После передачи условия СТАРТ аппаратно устанавливается флаг INT, а в регистр TWSR помещается код состояния $08 (см. Таблицу 88). Для перевода в режим ведущего передатчика необходимо передать ПОДЧИН_АДР + ЗАПИСЬ. Это выполняется путем записи значения ПОДЧИН_АДР + ЗАПИСЬ в регистр TWDR. После этого необходимо сбросить флаг TWINT (путем записи в него лог. 1) для продолжения сеанса связи. Данное выполняется путем записи следующего значения в TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	0	x	1	0	x

После передачи ПОДЧИН_АДР + ЗАПИСЬ и приема бита подтверждения флаг TWINT снова устанавливается, а в регистр TWSR помещается код состояния, который может иметь несколько значений. В режиме ведущего код состояния может быть $18, $20 или $38. Для каждого из этих кодов состояний необходимо выполнить адекватные действия, что отражено в таблице 66.

После успешной передачи ПОДЧИН_АДР + ЗАПИСЬ должен быть передан пакет данных. Его передача инициируется записью байта данных в TWDR. Доступ на запись к TWDR разрешен только тогда, когда флаг TWINT равен 1. В противном случае доступ блокируется и устанавливается флаг ошибочной записи TWWC в регистре TWCR. После обновления TWDR необходимо сбросить бит TWINT (путем записи в него лог. 1) для продолжения сеанса связи. Данное можно выполнить путем записи следующего значения в регистр TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	0	x	1	0	x

Данная последовательность повторяется до тех пор, пока не будет передан последний байт. После этого генерируется условие СТОП или ПОВТОРНЫЙ СТАРТ. Условие СТОП генерируется путем записи следующего значения TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	1	x	1	0	x

Условие ПОВТОРНЫЙ СТАРТ генерируется путем записи следующего значения в TWCR:

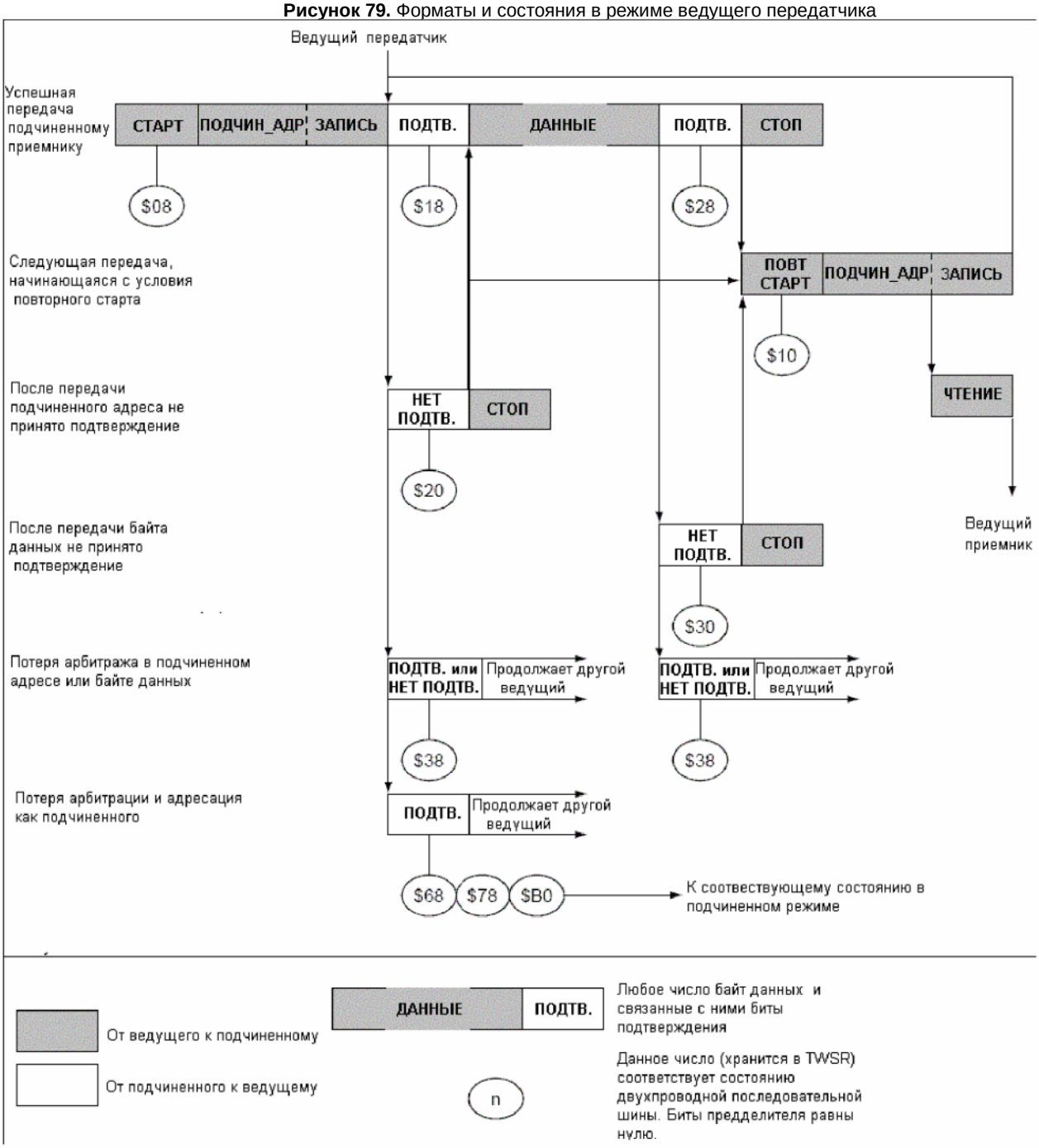
TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	1	0	x	1	0	x

После передачи условия ПОВТОРНОГО СТАРТА (состояние $10) двухпроводной последовательный интерфейс может обращаться к тому же подчиненному устройству или же к новому, при этом не требуется передача условия СТОП. Таким образом, повторный СТАРТ полезно использовать для смены подчиненного устройства в режимах ведущий передатчик и ведущий приемник без потери управления шиной.

Таблица 66. Коды состояния в режиме ведущего передатчика

Код состояния (TWSR), биты предделителя равны 0	Состояние двухпроводной последовательной шины и схемы двухпроводного последовательного интерфейса	Программные действия: Вниз TWDR	В TWCR: STA	STO	TWINT	TWEA	Следующее действие, выполняемое схемой TWI
$08	Передано условие СТАРТ	Загрузка ПОДЧИН_АДР+ЗАПИСЬ	0	0	1	x	Передается ПОДЧИН_АДР + ЗАПИСЬ Принимается ПОДТВ. или НЕТ ПОДТВ.
$10	Передано условие ПОВТОРНЫЙ СТАРТ	Загрузка ПОДЧИН_АДР+ЗАПИСЬ	0	0	1	X	Передается ПОДЧИН_АДР + ЗАПИСЬ; Принимается ПОДТВ или НЕТ ПОДТВ
		или ПОДЧИН_АДР+ЧТЕНИЕ	0	0	1	X	Передается ПОДЧИН_АДР + ЧТЕНИЕ; Переход на режим ведущего приемника
$18	Передано ПОДЧИН_АДР+ЗАПИСЬ и принято ПОДТВерждение	Загрузка байта данных или	0	0	1	X	Передается байт данных, принимается или не принимается ПОДТВерждение
		действия без загрузки TWDR или	1	0	1	X	Передается ПОВТОРНЫЙ СТАРТ
		действия без загрузки TWDR или	0	1	1	X	Передается условие СТОП и сбрасывается флаг TWSTO
		действия без загрузки TWDR	1	1	1	X	Вслед за условием СТАРТ передается условие СТОП и сбрасывается флаг TWSTO
$20	Передано ПОДЧИН_АДР+ЗАПИСЬ и принято НЕТ ПОДТВ	Загрузка байта данных или	0	0	1	X	Передается байт данных, принимается или не принимается ПОДТВерждение
		действия без загрузки TWDR или	1	0	1	X	Передается ПОВТОРНЫЙ СТАРТ
		действия без загрузки TWDR или	0	1	1	X	Передается условие СТОП и сбрасывается флаг TWSTO
		действия без загрузки TWDR	1	1	1	X	Вслед за условием СТАРТ передается условие СТОП и сбрасывается флаг TWSTO
$28	Передается байт данных; принимается ПОДТВерждение	Загрузка байта данных или	0	0	1	X	Передается байт данных, принимается или не принимается ПОДТВерждение
		действия без загрузки TWDR или	1	0	1	X	Передается ПОВТОРНЫЙ СТАРТ
		действия без загрузки TWDR или	0	1	1	X	Передается условие СТОП и сбрасывается флаг TWSTO
		действия без загрузки TWDR	1	1	1	X	Вслед за условием СТАРТ передается условие СТОП и сбрасывается флаг TWSTO
$30	Передается байт данных; принимается НЕТ ПОДТВерждения	Загрузка байта данных или	0	0	1	X	Передается байт данных, принимается или не принимается ПОДТВерждение
		действия без загрузки TWDR или	1	0	1	X	Передается ПОВТОРНЫЙ СТАРТ
		действия без загрузки TWDR или	0	1	1	X	Передается условие СТОП и сбрасывается флаг TWSTO
		действия без загрузки TWDR	1	1	1	X	Вслед за условием СТАРТ передается условие СТОП и сбрасывается флаг TWSTO
$38	Потеря арбитража при передачи ПОДЧИН_АДР + ЗАПИСЬ или байта данных	Нет действий с TWDR или	0	0	1	X	Освобождается двухпроводная шина и вводится неадресуемый подчиненный режим
		нет действий с TWDR	1	0	1	X	Передача условия СТАРТ после освобождения шины

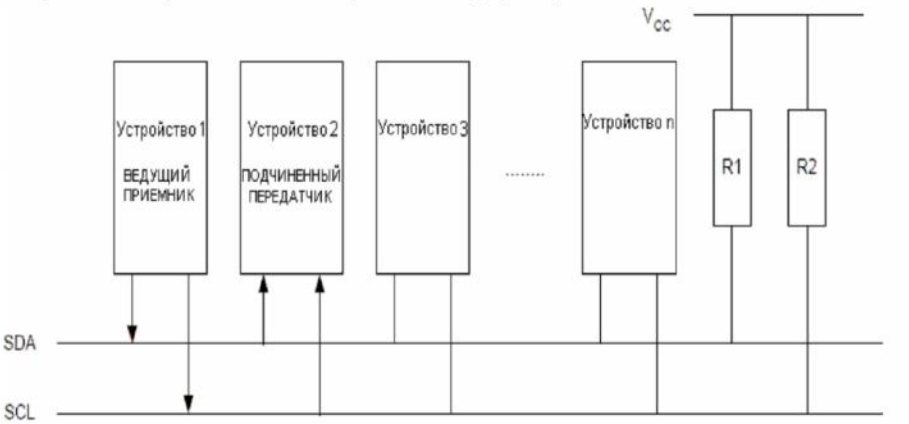
Рисунок 79. Форматы и состояния в режиме ведущего передатчика



Режим ведущего приемника

В режиме ведущего приемника принимается несколько байт данных от подчиненного приемника (см. рисунок 98). Для ввода режима ведущего необходимо передать условие СТАРТ. Формат следующего адресного пакета определит, будет ли введенный режим ведущий передатчик или приемник. Если передается ПОДЧИН_АДР+ЗАПИСЬ, то вводится режим ведущий передатчик, если же передается ПОДЧИН_АДР+ЧТЕНИЕ, то вводится режим ведущий приемник. Во всех кодах состояния, приведенных в этом разделе, не учитываются биты предделителя и они равны нулю.

Рисунок 80. Передача данных в режиме ведущего приемника



Передача условия СТАРТ инициируется путем записи следующего значения в TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	1	0	x	1	0	x

Для разрешения работы двухпроводного последовательного интерфейса необходимо установить бит TWEN. Передача условия СТАРТ инициируется записью лог. 1 в TWSTA. Для сброса флага TWINT необходимо записать в него лог. 1. TWI выполнит генерацию условия СТАРТ только после тестирования шины и определения ее освобождения. После передачи условия СТАРТ флаг TWINT устанавливается аппаратно, а в регистр TWSR помещается код состояния $08 (см. таблицу 66). Для ввода режима ведущий приемник необходимо передать ПОДЧИН_АДР+ЧТЕНИЕ. Данное выполняется путем записи значения ПОДЧИН_АДР+ЧТЕНИЕ в TWDR. После этого необходимо сбросить флаг TWINT (путем записи в него лог. 1) для продолжения сеанса связи. Для этого в регистр TWCR необходимо поместить следующее значение:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	0	x	1	0	x

После передачи ПОДЧИН_АДР+ЧТЕНИЕ и приема бита подтверждения устанавливается снова флаг TWINT, а в регистр TWSR помещается код состояния, который может иметь несколько значений: $38, $40 или $48. Действия, которые выполняются при каждом из этих значений представлены в таблице 67. Принятые данные хранятся в регистре TWDR после аппаратной установки флага TWINT. Данная последовательность повторяется до приема последнего байта. После этого ведущий приемник информирует подчиненный передатчик отправкой НЕТ ПОДТВерждения после приема последнего принятого байта данных. Сеанс связи завершается генерацией условия СТОП или ПОВТОРНЫЙ СТАРТ. Условие СТОП генерируется путем записи в регистр TWCR следующего значения:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	1	x	1	0	x

Условие ПОВТОРНЫЙ СТАРТ генерируется путем записи в TWCR следующего значения:

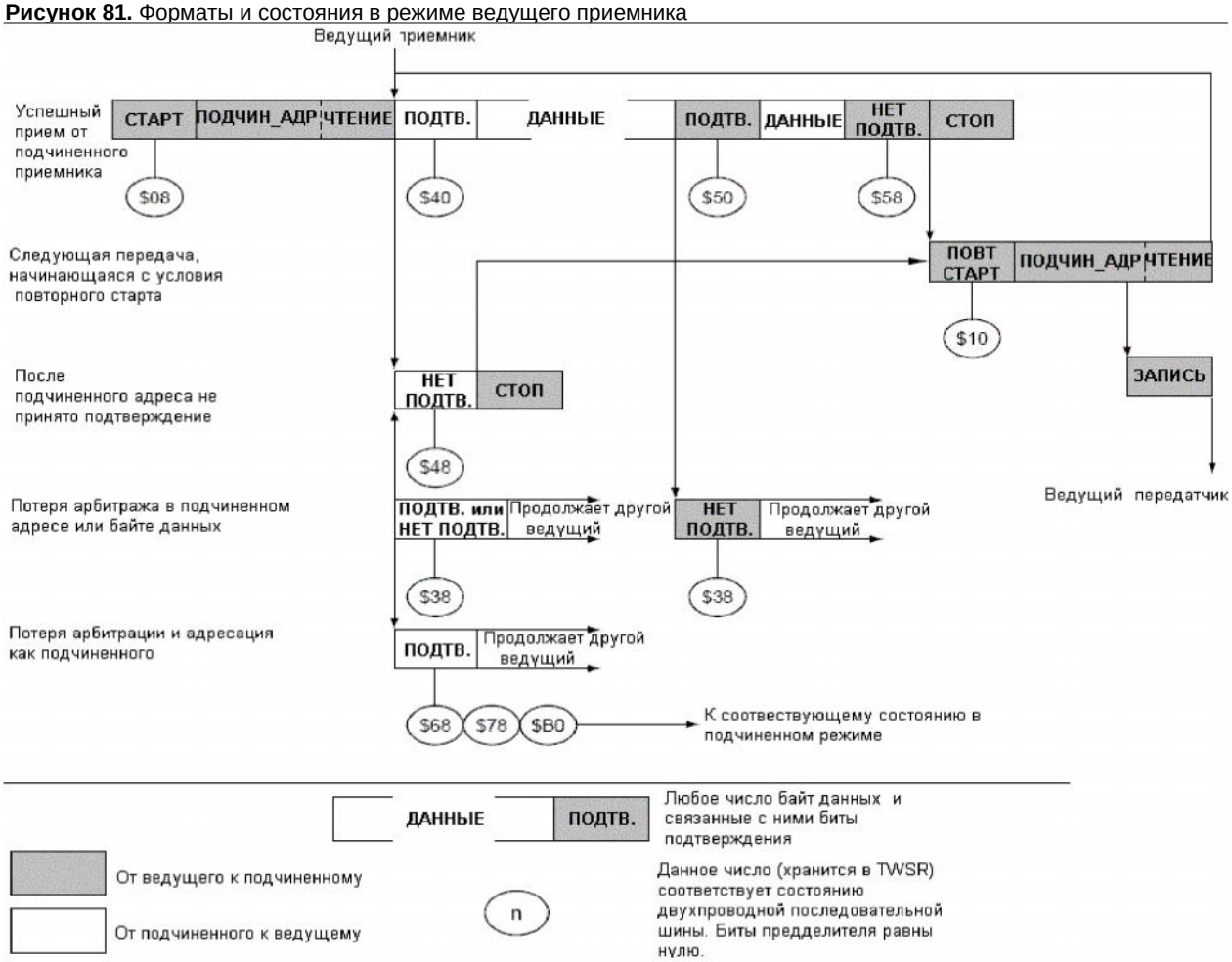
TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	1	0	x	1	0	x

После генерации условия ПОВТОРНЫЙ СТАРТ (состояние $10) двухпроводной последовательный интерфейс может обращаться к тому же подчиненному или к новому подчиненному без генерации условия СТОП. ПОВТОРНЫЙ СТАРТ позволяет ведущему переключаться между подчиненными, режимом ведущего передатчика и ведущего приемника без потери управления над шиной.

Таблица 67. Коды состояния для режима ведущего приемника

Код состояния (TWSR), биты предделителя равны 0	Состояние двухпроводной последовательной шины и схемы двухпроводного последовательного интерфейса	Программные действия					Следующее действие, выполняемое схемой TWI
		Вклз TWDR	В TWCR				
			STA	STO	TWINT	TWEA	
$08	Передано условие СТАРТ	Загрузка ПОДЧИН_АДР+ЧТЕНИЕ	0	0	1	x	Передается ПОДЧИН_АДР + ЧТЕНИЕ Принимается ПОДТВ. или НЕТ ПОДТВ.
$10	Передано условие ПОВТОРНЫЙ СТАРТ	Загрузка ПОДЧИН_АДР+ЧТЕНИЕ	0	0	1	X	Передается ПОДЧИН_АДР+ЧТЕНИЕ, принимается ПОДТВ. или НЕТ ПОДТВ.
		или ПОДЧИН_АДР+ЗАПИСЬ	0	0	1	X	Передается ПОДЧИН_АДР+ЗАПИСЬ, переключение на режим «Ведущий передатчик»
$38	Потеря арбитрации во время передачи бит ПОДЧИН_АДР + ЧТЕНИЕ или бита НЕТ ПОДТВ.	Действия без загрузки TWDR или	0	0	1	X	Шина освобождается и вводится безадресный подчиненный режим
		действия без загрузки TWDR	1	0	1	X	Условие СТАРТ передается после освобождения шины
$40	Передано ПОДЧИН_АДР+ЧТЕНИЕ и принято ПОДТВ	Действия без загрузки TWDR или	0	0	1	0	Принимается байт данных, возвращается бит НЕТ_ПОДТВ.
		действия без загрузки TWDR	0	0	1	1	Принимается байт данных, возвращается бит ПОДТВ.
$48	Передано ПОДЧИН_АДР+ЧТЕНИЕ и принято НЕТ ПОДТВ	Действия без загрузки TWDR или	1	0	1	X	Передается ПОВТОРНЫЙ СТАРТ
		действия без загрузки TWDR или	0	1	1	X	Передается условие СТОП и сбрасывается флаг TWSTO
		действия без загрузки TWDR	1	1	1	X	Вслед за условием СТАРТ передается условие СТОП и сбрасывается флаг TWSTO
$50	Принят байт данных и возвращается ПОДТВерждение	Чтение байта данных или	0	0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВ
		чтение байта данных	0	0	1	1	Принимается байт данных и возвращается ПОДТВ
$58	Принят байт данных и возвращается НЕТ ПОДТВерждения	Чтение байта данных или	1	0	1	X	Передается ПОВТОРНЫЙ СТАРТ
		чтение байта данных или	0	1	1	X	Передается СТОП и сбрасывается флаг TWSTO
		чтение байта данных	1	1	1	X	Вслед за условием СТАРТ передается условие СТОП и сбрасывается флаг TWSTO

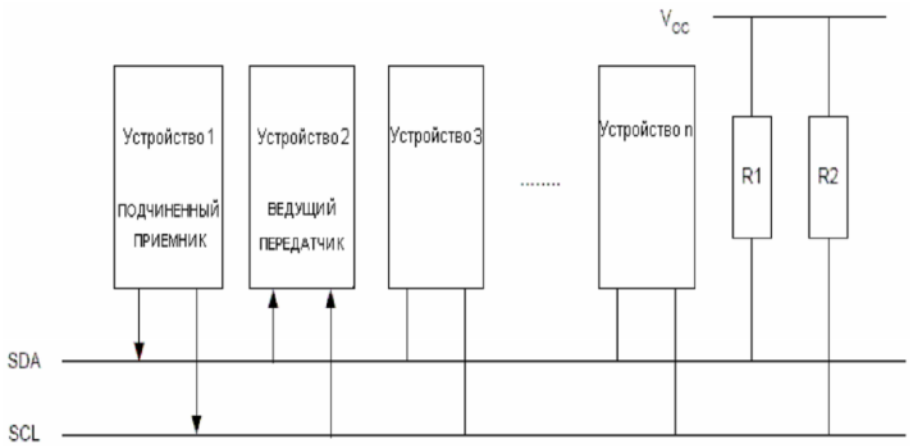
Рисунок 81. Форматы и состояния в режиме ведущего приемника



Режим подчиненного приемника

В режиме подчиненного приемника принимается несколько байт данных от ведущего передатчика (см. рисунок 82). Во всех кодах состояния, приведенных в этом разделе, не учитываются биты предделителя и они равны нулю.

Рисунок 82. Передача данных в режиме подчиненного приемника



Для ввода режима подчиненного приемника необходимо выполнить инициализацию регистров TWAR и TWCR следующим образом:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Значение	Собственный подчиненный адрес устройства							

Старшие семь разрядов образуют адрес. С помощью него определяется к какому двухпроводному интерфейсу адресуется мастер. Если в младшем разряде записана лог. 1, то TWI будет отвечать на адрес общего вызова ($00). В противном случае он игнорирует адрес общего вызова.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	0	1	0	0	0	1	0	x

Для разрешения работы TWI необходимо записать лог. 1 в TWEN. Для разрешения подтверждения собственно подчиненного адреса или адреса общего вызова записывается 1 в TWEA. Битам TWSTA и TWSTO необходимо присвоить нулевое значение.

После инициализации TWAR и TWCR схема TWI ожидает получения собственного подчиненного адреса (или, если разрешено, адреса общего вызова), а вслед за ним – бита направления данных. Если бит направления равен "0" (запись), то TWI переходит в режим "подчиненный приемник", в противном случае вводится режим "подчиненный передатчик". После приема собственного подчиненного адреса и бита записи устанавливается флаг TWINT, а в регистр TWSR помещается код состояния. По коду состояния определяется какие программные действия необходимо предпринять. В таблице 90 собрана информация о предпринимаемых действиях для каждого возможного значения кода состояния. Режим подчиненного приемника также вводится, если теряется арбитрация, когда TWI находился в режиме ведущего (см. состояния $68 и $78).

Если бит TWEA сбросить во время передачи, то TWI ответит "НЕТ ПОДТВ " ("1") на линии SDA после приема следующего байта данных. Данное свойство может использоваться для сигнализации состояния, когда подчиненный не может больше принимать байты данных. Если TWEA равен нулю, то TWI не подтверждает свой подчиненный адрес. Однако последовательная шина остается под контролем и функция распознавания адреса может быть активизирована в любой момент путем установки бита TWEA. Это означает, что бит TWEA можно использовать для временной изоляции TWI от двухпроводной последовательной шины.

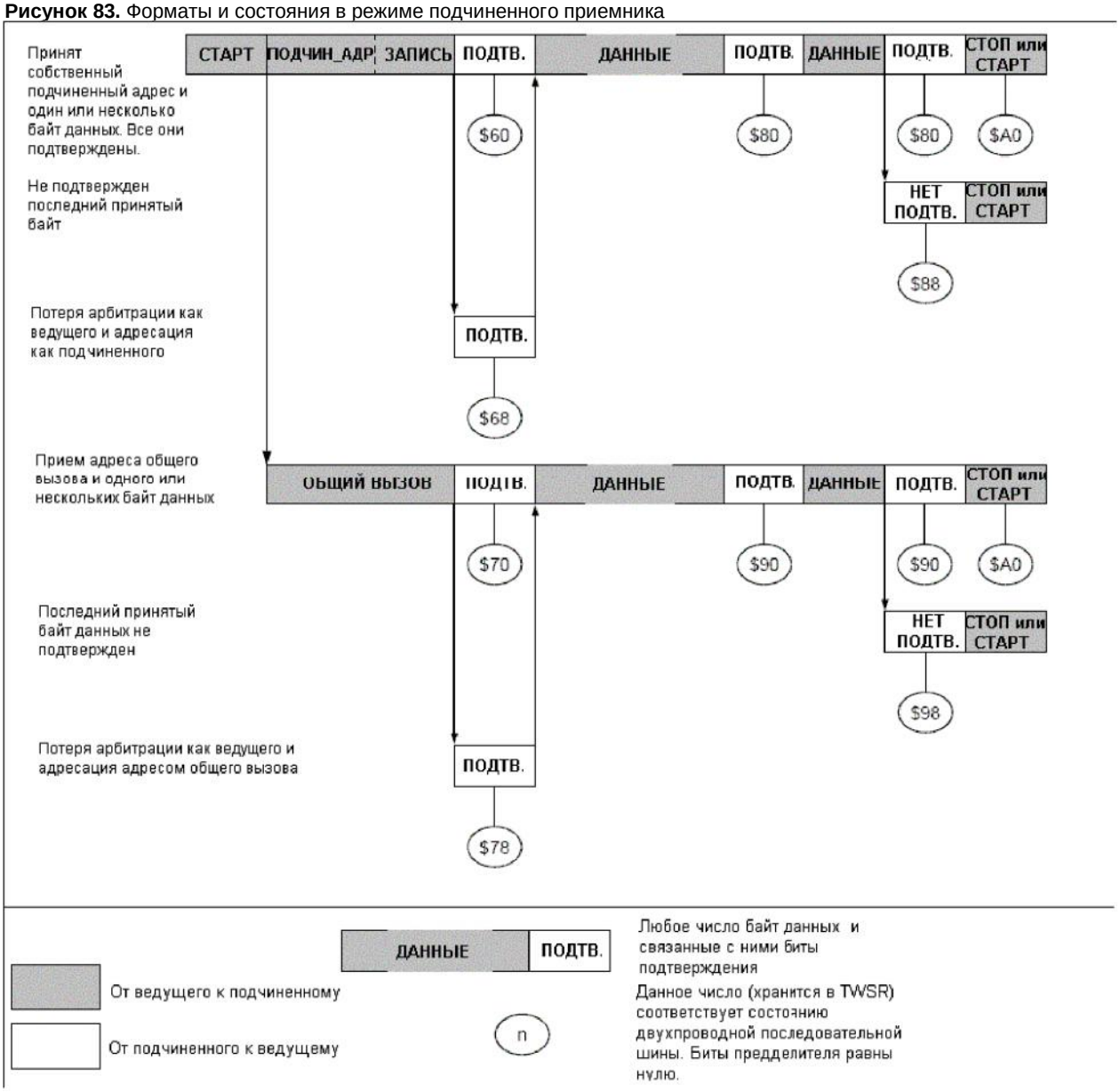
Во всех режимах сна, кроме холостого хода (Idle), синхронизация TWI отключается. Если бит TWEA установлен, то интерфейс останется способным подтверждать прием своего собственного подчиненного адреса или адреса общего вызова за счет использования сигнала синхронизации шины в качестве тактового источника. При обнаружении запроса микроконтроллер выходит из режима сна, при этом линия SCL остается на низком уровне в процессе пробуждения и до сброса флага TWINT (записью в него "1"). Далее выполняется прием данных обычным способом при обычной системной синхронизации. Учтите, что если для микроконтроллера выбрано большое время запуска, то линия SCL может оказаться длительно в низком состоянии и заблокировать другой обмен информацией.

Обратите внимание, что после пробуждения регистр данных TWDR не отражает последний байт, присутствовавший на шине во время выхода из указанных выше режимов сна.

Таблица 68. Коды состояния для режима "Подчиненный приемник"

Код состояния (TWSR), биты предделителя равны 0	Состояние двухпроводной последовательной шины и схемы двухпроводного последовательного интерфейса	Программные действия					Следующее действие, выполняемое схемой TWI
		Из TWDR	В TWCR				
			STA	STO	TWINT	TWEA	
$60	Принимается собственный ПОДЧИН_АДР+ЗАПИСЬ; возвращено ПОДТВ.	Действия без загрузки TWDR или	x	0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВерждение
		действия без загрузки TWDR	x	0	1	1	Принимается байт данных и возвращается ПОДТВерждение
$68	Потеряна арбитрация во время передачи ПОДЧИН_АДР+ЧТЕНИЕ/ ЗАПИСЬ как ведущего, принят собственный ПОДЧИН_АДР+ЗАПИСЬ, возвращено ПОДТВ	Действия без загрузки TWDR или	x	0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВерждение
		действия без загрузки TWDR	x	0	1	1	Принимается байт данных и возвращается ПОДТВерждение
$70	Принят адрес общего вызова; возвращено подтверждение	Действия без загрузки TWDR или	x	0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВерждение
		действия без загрузки TWDR	x	0	1	1	Принимается байт данных и возвращается ПОДТВерждение
$78	Потеряна арбитрация во время передачи ПОДЧИН_АДР+ЧТЕНИЕ/ ЗАПИСЬ как ведущего, принят адрес общего вызова, возвращено ПОДТВ	Действия без загрузки TWDR или	x	0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВерждение
		действия без загрузки TWDR	x	0	1	1	Принимается байт данных и возвращается ПОДТВерждение
$80	Предварительная адресация собственным ПОДЧИН_АДР+ЗАПИСЬ; приняты данные; возвращено ПОДТВерждения	Чтение байта данных или	x	0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВерждение
		чтение байта данных	x	0	1	1	Принимается байт данных и возвращается ПОДТВерждение
$88	Предварительная адресация собственным ПОДЧИН_АДР+ЗАПИСЬ; приняты данные; возвращено НЕТ ПОДТВерждения	Чтение байта данных или	0	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова
		чтение байта данных или	0	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"
		чтение байта данных или	1	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова; передается условие СТАРТ после освобождения шины
		чтение байта данных	1	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"; передается условие СТАРТ после освобождения шины
$90	Предварительная адресация адресом общего вызова; приняты данные; возвращено ПОДТВерждение	Чтение байта данных или	x	0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВерждение
		чтение байта данных	x	0	1	1	Принимается байт данных и возвращается ПОДТВерждение
$98	Предварительная адресация адресом общего вызова; приняты данные; возвращено НЕТ ПОДТВерждения	Чтение байта данных или	0	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова
		чтение байта данных или	0	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"
		чтение байта данных или	1	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова; передается условие СТАРТ после освобождения шины
		чтение байта данных	1	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"; передается условие СТАРТ после освобождения шины
$A0	Принято условие СТОП или повторный СТАРТ во время подчиненной адресации	Нет действий	0	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова
			0	0	1	1	Переключение на безадресный

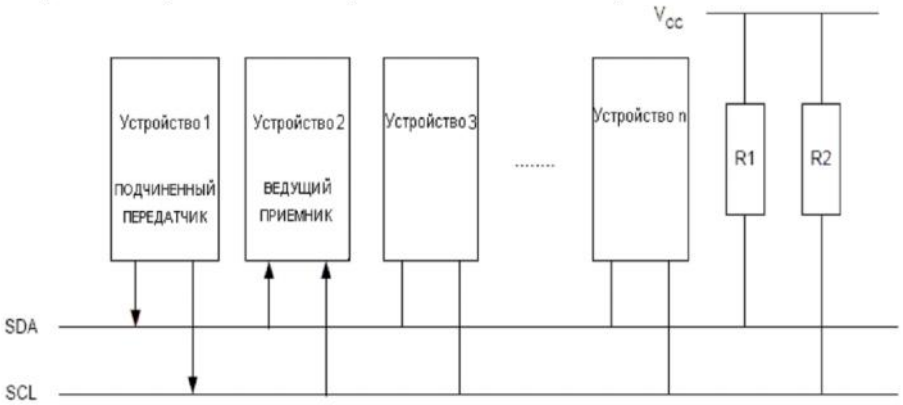
Рисунок 83. Форматы и состояния в режиме подчиненного приемника



Режим подчиненного передатчика

В режиме подчиненного передатчика выполняется передача нескольких байт данных ведущему приемнику (см. рисунок 84). Во всех кодах состояния, приведенных в этом разделе, не учитываются биты предделителя и они равны нулю.

Рисунок 84. Передача данных в режиме подчиненного передатчика



Для ввода режима подчиненного передатчика необходимо инициализировать регистры TWAR и TWCR следующим образом:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Значение	Собственный подчиненный адрес устройства							

Старшие семь разрядов образуют адрес. С помощью него определяется к какому двухпроводному интерфейсу адресуется ведущий. Если в младшем разряде записана лог. 1, то TWI будет отвечать на адрес общего вызова ($00). В противном случае он игнорирует адрес общего вызова.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	0	1	0	0	0	1	0	x

Для разрешения работы TWI необходимо записать лог. 1 в TWEN. Для разрешения подтверждения собственного подчиненного адреса или адреса общего вызова записывается 1 в TWEA. Битам TWSTA и TWSTO необходимо присвоить нулевое значение.

После инициализации TWAR и TWCR схема TWI ожидает получения собственного подчиненного адреса (или, если разрешен, адреса общего вызова), а вслед за ним – бита направления данных. Если бит направления равен "1" (чтение), то TWI переходит в режим "подчиненный передатчик", иначе вводится режим "подчиненный приемник". После приема собственно подчиненного адреса и бита записи устанавливается флаг TWINT, а в регистр TWSR помещается код состояния. Код состояния позволяет определить, какие программные действия необходимо выполнить. Подробности по использованию кодов состояний представлены в таблице 69. Режим "подчиненный передатчик" также вводится, если теряется арбитрация, когда TWI находился в режиме ведущего (см. состояние $B0).

Если бит TWEA обнулить во время передачи, то TWI передаст последний байт. Вводится состояние $C0 или состояние $C8 в зависимости от того принял ПОДТВ или НЕТ ПОДТВ ведущий приемник за последним байтом. TWI переходит в безадресный подчиненный режим и далее игнорирует ведущего, если тот продолжает передачу. Таким образом, ведущий приемник принимает все "1" как последовательные данные. Состояние $C8 вводится, если ведущий требует передачи дополнительных байт данных (путем передачи ПОДТВ), даже если подчиненный передал последний байт (TWEA равен нулю и ожидается прием НЕТ ПОДТВ от ведущего).

Пока TWEA равен нулю, TWI не отвечает на собственный подчиненный адрес. Однако последовательная шина остается под контролем и функция распознавания адреса может быть активизирована в любой момент путем установки бита TWEA. Это означает, что бит TWEA можно использовать для временной изоляции TWI от двухпроводной последовательной шины.

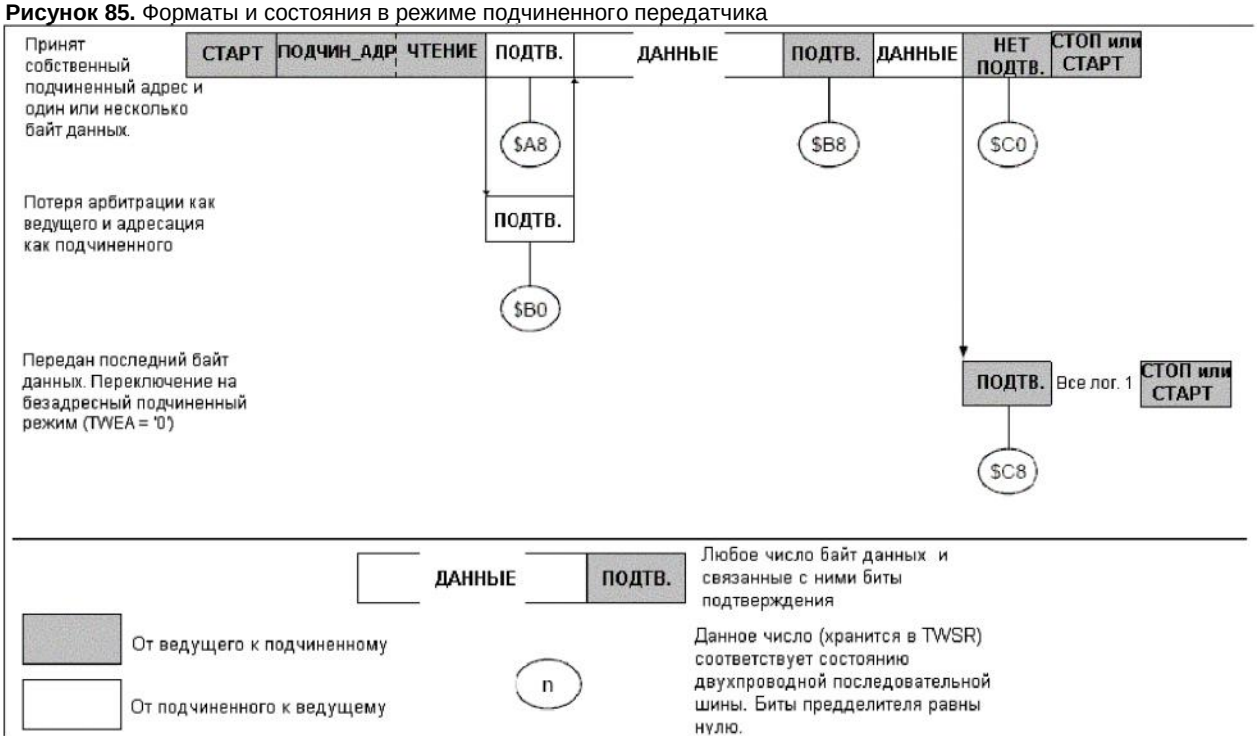
Во всех режимах сна, кроме холостого хода (Idle), синхронизация TWI отключается. Если бит TWEA установлен, то интерфейс останется способным подтверждать прием своего собственного подчиненного адреса или адреса общего вызова за счет использования сигнала синхронизации шины в качестве тактового источника. При обнаружении запроса микроконтроллер выходит из режима сна, при этом линия SCL остается на низком уровне в процессе пробуждения и до сброса флага TWINT (записью в него "1"). Далее выполняется прием данных обычным способом при обычной системной синхронизации. Учтите, что если для микроконтроллера выбрано большое время запуска, то линия SCL может оказаться длительно в низком состоянии и заблокировать другой обмен информацией.

Обратите внимание, что после пробуждения регистр данных TWDR не отражает последний байт, присутствовавший на шине во время выхода из указанных выше режимов сна.

Таблица 69. Коды состояния для режима "Подчиненный передатчик"

Код состояния (TWSR), биты предделителя равны 0	Состояние двухпроводной последовательной шины и схемы двухпроводного последовательного интерфейса	Программные действия					Следующее действие, выполняемое схемой TWI
		В/из TWDR	В TWCR				
			STA	STO	TWINT	TWEA	
$A8	Принимается собственный ПОДЧИН_АДР+ЧТЕНИЕ; возвращено ПОДТВ.	Загрузка байта данных или	x	0	1	0	Передается последний байт данных и должно быть принято НЕТ ПОДТВ
		загрузка байта данных	x	0	1	1	Передается байт данных и должно быть принято ПОДТВ
$B0	Потеряна арбитрация во время передачи ПОДЧИН_АДР+ЧТЕНИЕ/ ЗАПИСЬ как ведущего, принят собственный ПОДЧИН_АДР+ЧТЕНИЕ, возвращено ПОДТВ	Загрузка байта данных или	x	0	1	0	Передается последний байт данных и должно быть принято НЕТ ПОДТВ
		загрузка байта данных	x	0	1	1	Передается байт данных и должно быть принято ПОДТВ
$B8	Передан байт данных из TWDR; принято ПОДТВерждение	Загрузка байта данных или	x	0	1	0	Передается последний байт данных и должно быть принято НЕТ ПОДТВ
		загрузка байта данных	x	0	1	1	Передается байт данных и должно быть принято ПОДТВ
$C0	Передан байт данных из TWDR; принято НЕТ ПОДТВерждения	Действия без загрузки TWDR или действия без загрузки TWDR или действия без загрузки TWDR или действия без загрузки TWDR	0	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова
			0	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"
			1	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова, передается условие СТАРТ после освобождения шины
			1	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"; передается условие СТАРТ после освобождения шины
$C8	Передан последний байт данных из TWDR (TWEA = "0"); принято ПОДТВерждение	Действия без загрузки TWDR или действия без загрузки TWDR или действия без загрузки TWDR или действия без загрузки TWDR	0	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова
			0	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"
			1	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова; передается условие СТАРТ после освобождения шины
			1	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"; передается условие СТАРТ после освобождения шины

Рисунок 85. Форматы и состояния в режиме подчиненного передатчика



Прочие состояния

Имеются несколько кодов состояний, которые отличаются от упомянутых выше (см. табл. 70). Состояние $F8 индицирует, что нет доступной информации, т.к. не установлен флаг TWINT. Это может произойти между другими состояниями и когда TWI не участвует в последовательной передаче данных.

Состояние $00 индицирует, что во время последовательной передачи данных на шине возникла ошибка. Ошибка возникает, если условия СТАРТ или СТОП возникают в неверной позиции формата посылки. Примеры таких неточных позиций могут существовать во время передачи адресного байта, байта данных и бита подтверждения. После возникновения ошибки устанавливается флаг TWINT. Для выхода из состояния ошибки необходимо установить флаг TWSTO и сбросить TWINT путем записи в него "1". Это приводит к переводу TWI в безадресный режим и к сбросу флага TWSTO (другие биты в TWCR не затрагиваются). Линии SDA и SCL освобождаются и условие СТОП не передается.

Таблица 70. Прочие состояния

Код состояния (TWSR) Биты Делителя частоты 0	Состояние TWI и двухпроводных аппаратных средств	Программные установки					Действия, предпринятые Аппаратными средствами TWI
		К/от TWDR	К TWCR				
			STA	STO	TWINT	TWEA	
0xF8	Нет доступной информации о состоянии; TWINT = “0”	TWDR не действует	TWDR не действует				Ожидание или продолжение текущей передачи
0x00	Ошибка шины из-за недопустимого условия СТАРТа или СТОПа	TWDR не действует	0	1	1	x	Влияет только на внутренние аппаратные средства, никакое условие СТОПа не отправляется по шине. Во всех случаях шина свободна, и TWSTO очищается.

Сочетание нескольких режимов

В некоторых случаях сочетаются несколько режимов TWI для обеспечения желаемого действия. В качестве примера рассмотрим чтение данных из последовательного EEPROM. Обычно, такой сеанс связи организовывается в такой последовательности:
1. Инициируется сеанс связи
2. В EEPROM отправляется инструкция с указанием адреса считываемой ячейки
3. Выполняется чтение
4. Завершается сеанс связи

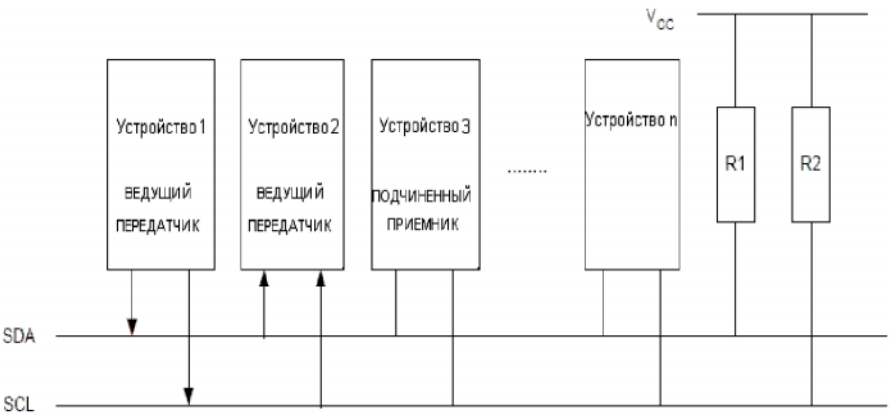
Обратите внимание, что данные передаются как от ведущего к подчиненному, так и обратно, от подчиненного к ведущему. Ведущий инструктирует подчиненного какую ячейку он желает считать, для чего используется режим "Ведущий передатчик". В дальнейшем данные передаются подчиненным, что требует использования режима "Ведущий приемник". Следовательно, направление передачи данных изменяется. Ведущий должен сохранить управление над шиной на каждом из этапов, а каждый из шагов должен быть выполнен как элементарное действие. Если данный принцип нарушить в многомастерной системе, то другой ведущий может обратиться к EEPROM на шагах 2 и 3 и изменить указатель данных. Это приведет к тому, что ведущий считывает данные из ячейки с неверным адресом. Таким образом, направление передачи данных необходимо изменять только передачей условия ПОВТОРНЫЙ СТАРТ между передачей адресного байта и приемом байта. Передачей ПОВТОРНОГО СТАРТА мастер сохранит свое "господство" на шине. На следующем рисунке представлена структура потока данной передачи.

Рисунок 86. Сочетание нескольких режимов TWI для обмена данными с последовательным EEPROM

Мульти-ведущие системы и арбитраж

Если к одной шине подключено несколько ведущих, то передача может быть инициирована одновременно одним или несколькими из них. Стандарт TWI гарантирует, что в таких ситуациях разрешается передача только одному ведущему, при этом не будет происходить потеря данных. Пример арбитрирования такой ситуации представлен ниже, где два ведущих пытаются передавать данные подчиненному приемнику.

Рисунок 87. Пример арбитрации



Ниже приведены несколько различных сценариев, возникающих в процессе арбитрирования:

• Два или более ведущих выполняют идентичную связь с одним и тем же подчиненным. В этом случае ни один подчиненный и ни один из ведущих не узнает об этой конфликтной ситуации.

• Два или более ведущих обращаются к тому же подчиненному с различными данными или битом направления данных. В этом случае возникает арбитрирование во время передачи бита ЧТЕНИЕ/ЗАПИСЬ или же бит данных. Одни ведущие выводят на SDA лог. 1, а другие выводят лог.0. Последние теряют арбитрацию. Ведущие, которые проиграли арбитрирование, переходят в безадресный подчиненный режим или ожидают освобождения шины и затем передают новое условие СТАРТ, что зависит от программных действий.

• Два или более ведущих обращаются к различным подчиненным. В этом случае арбитрирование возникает во время передачи бит ПОДЧИН_АДР. Одни ведущие выводят на SDA лог. 1, а другие выводят лог.0. Последние теряют арбитрацию. Ведущие, которые проиграли арбитрирование во время передачи ПОДЧИН_АДР, переходят в подчиненный режим для проверки не обращается ли к ним выигравший арбитраж ведущий. Если такая адресация действительно выполняется, то они переключаются в режим "Подчиненный приемник" или "Подчиненный передатчик" в зависимости от значения бита ЧТЕНИЕ/ЗАПИСЬ. Если адресации не было, то они перейдут в безадресный подчиненный режим или будут ожидать освобождения шины и после этого передадут новое условие СТАРТ (задается программно).

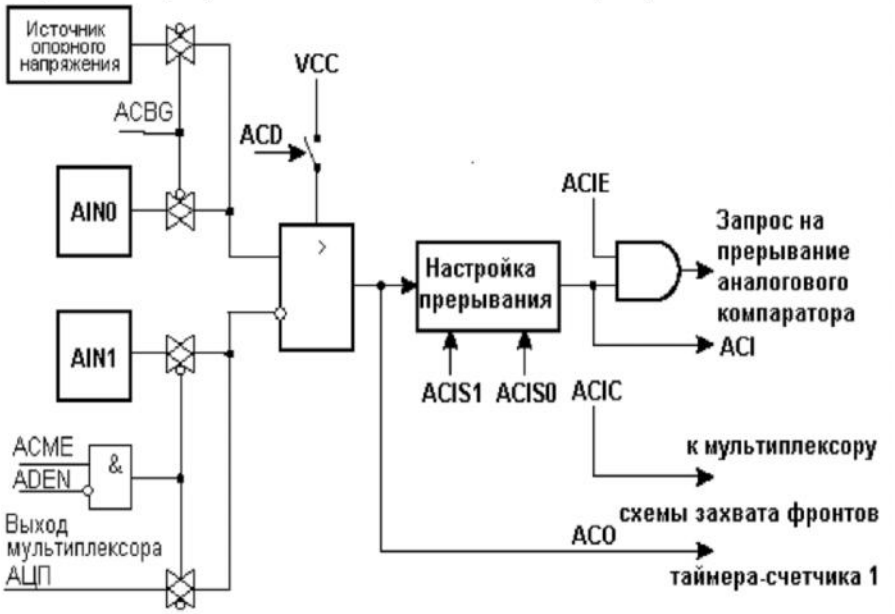
Данные действия обобщены на рисунке 88. Возможные коды состояний представлены внутри окружностей.

Рисунок 88. Возможные коды состояний при потере арбитрации

Аналоговый компаратор

Аналоговый компаратор сравнивает уровни напряжений на неинвертирующем входе AIN0 и инвертирующем входе AIN1. Если напряжение на неинвертирующем входе AIN0 превышает напряжение на инвертирующем входе AIN1, то выход аналогового компаратора ACO принимает единичное состояние. Выход компаратора может быть настроен для использования в качестве источника входного сигнала для схемы захвата фронтов таймера-счетчика 1. Кроме того, компаратор может генерировать собственный запрос на обработку прерывания. Пользователь может выбрать несколько событий, по которым возникает прерывание: нарастающий, падающий фронт на выходе компаратора или любое его изменение. Функциональная схема компаратора и связанной с ним логики представлена на рисунке 89.

Рисунок 89. Функциональная схема аналогового компаратора



Прим. 1: См. табл. 72 стр.195.
Прим. 2: Информация по расположению выводов аналогового компаратора представлена на стр. 2 и в таблице 28 стр63.

Регистр специальных функций ввода-вывода – SFIOR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 3 – ACME: Выбор мультиплексора на входе аналогового компаратора
Если выключен аналогово-цифровой преобразователь (ADEN=0 в регистре ADCSRA) и в данный разряд записана лог. 1, то к инвертирующему входу аналогового компаратора подключен выход аналогового мультиплексора АЦП. Запись в данный разряд лог. 0 приведет к подключению инвертирующего входа аналогового компаратора к выводу микроконтроллера AIN1. Более подробно об использовании данной настройки см. в разделе “Мультиплексированный вход аналогового компаратора”стр.195.

Регистр состояния и управления аналогового компаратора – ACSR

Bit	7	6	5	4	3	2	1	0	
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

• Bit 7 – ACD: Отключение аналогового компаратора
Запись в данный разряд лог. 1 приводит к снятию питания с аналогового компаратора. Данный разряд можно устанавливать в любой момент при необходимости отключения аналогового компаратора. Его использование позволяет снизить энергопотребление в активном режиме и режиме холостого хода. Перед изменением бита ACD необходимо отключить прерывание по аналоговому компаратору путем сброса бита ACIE в регистре ACSR. В противном случае может возникнуть прерывание после изменения значения данного бита.

• Bit 6 – ACBG: Подключение источника опорного напряжения к аналоговому компаратору
После установки данного бита к неинвертирующему входу компаратора подключается источник опорного напряжения. После сброса данного разряда неинвертирующий вход компаратора связан с выводом AIN0 микроконтроллера. См. также “Встроенный источник опорного напряжения”стр.42.

• Bit 5 – ACO: Выход аналогового компаратора
Данный бит выхода аналогового компаратора связан непосредственно с выходом ACO через цепь синхронизации. Синхронизация реализована как временная задержка на 1 – 2 машинных цикла.

• Bit 4 – ACI: Флаг прерывания аналогового компаратора
Данный разряд устанавливается аппаратно, при возникновении события в соответствии с установками бит ACIS1 и ACIS0. Запрос на обработку прерывания аналогового компаратора выполняется, если установлены биты ACIE и I в регистре SREG. ACI сбрасывается аппаратно при переходе на соответствующий вектор обработки прерывания. Альтернативно, бит ACI можно сбросить программно путем записи лог. 1 в данный флаг.

• Bit 3 – ACIE: Разрешение прерывания аналогового компаратора
Если в данный разряд записана лог. 1 и установлен бит I в регистре статуса, то прерывание по аналоговому компаратору активизируется. Запись в данный разряд лог. 0 приводит к отключению данного прерывания.

• Bit 2 – ACIC: Подключение аналогового компаратора к схеме захвата фронтов
Установка данного разряда приводит к разрешению совместной работы схемы захвата фронтов таймера-счетчика 1 и аналогового компаратора. В этом случае, выход аналогового компаратора непосредственно подключен к входному каскаду схемы захвата фронтов, позволяя к компаратору добавить функции подавления шумов и настройки фронтов прерывания по захвату фронта таймером-счетчиком 1.После записи в данный разряд лог. 0 связь между аналоговым компаратором и схемой захвата фронтов разрывается. Для активизации прерывания схемы захвата фронтов таймера-счетчика 1 по срабатыванию аналогового компаратора необходимо установить бит TICIE1 в регистре маски прерывания таймера (TIMSK).

• Bits 1,0 – ACIS1, ACIS0: Выбор события прерывания аналогового компаратора
Данные разряды определяют какое событие приводит к генерации запроса на прерывание аналогового компаратора. Варианты установок данных разрядов и их назначение представлены в табл. 71.
Таблица 71. Установки разрядов ACIS1, ACIS0

ACIS1	ACIS0	Событие
0	0	Прерывание по любому изменению на выходе компаратора
0	1	Зарезервировано
1	0	Прерывание по падающему фронту на выходе компаратора
1	1	Прерывание по нарастающему фронту на выходе компаратора

Перед изменением бит ACIS1/ACIS0 необходимо отключить прерывание по аналоговому компаратору путем сброса бита разрешения прерывания в регистре ACSR. В противном случае может возникнуть прерывание при изменении значений данных бит.

Мультиплексированный вход аналогового компаратора

Имеется возможность использовать выводы ADC7..0 в качестве неинвертирующих входов аналогового компаратора. Для организации такого ввода используется мультиплексор АЦП и, следовательно, в этом случае АЦП должен быть отключен. Если установлен бит разрешения подключения мультиплексора к аналоговому компаратору (бит ACME в SFIOR) и выключен АЦП (ADEN=0 в регистре ADCSRA), то состояние разрядов MUX2..0 регистра ADMUX определяют какой вывод микроконтроллера подключен к неинвертирующему входу аналогового компаратора (см. табл. 72). Если ACME сброшен или установлен ADEN, то в качестве неинвертирующего входа аналогового компаратора используется вывод микроконтроллера AIN1.

Таблица 72 . Мультиплексированный вход аналогового компаратора

ACME	ADEN	MUX2..0	Неинвертирующий вход аналогового компаратора
0	x	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

Прим.: 1. ADC7.. 6 только доступны в TQFP и QFN/MLF корпусах.

Аналогово-цифровой преобразователь

Техническая Характеристика

- **Разрешение 10-бит**
- **Нелинейность ±0.5 LSB**
- **Абсолютная Погрешность ± 2 LSB**
- **Время Преобразования 13 - 260 µs**
- **Не более 15 kSPS при Максимальном Разрешении**
- **Один из 6 Переключаемых Каналов**
- **Еще 2 Канала (TQFP and MLF Package only)**
- **Возможность выравнивания результата в левую сторону**
- **0 - VCC ADC Диапазон входного напряжения**
- **Подключаемое 2.56V ADC Опорное Напряжение**
- **Режимы Непрерывного (Free Running) и Единичного преобразования (Single Conversion)**
- **Прерывание по окончании преобразования ADC**
- **Шумоподавитель в Спящем режиме**

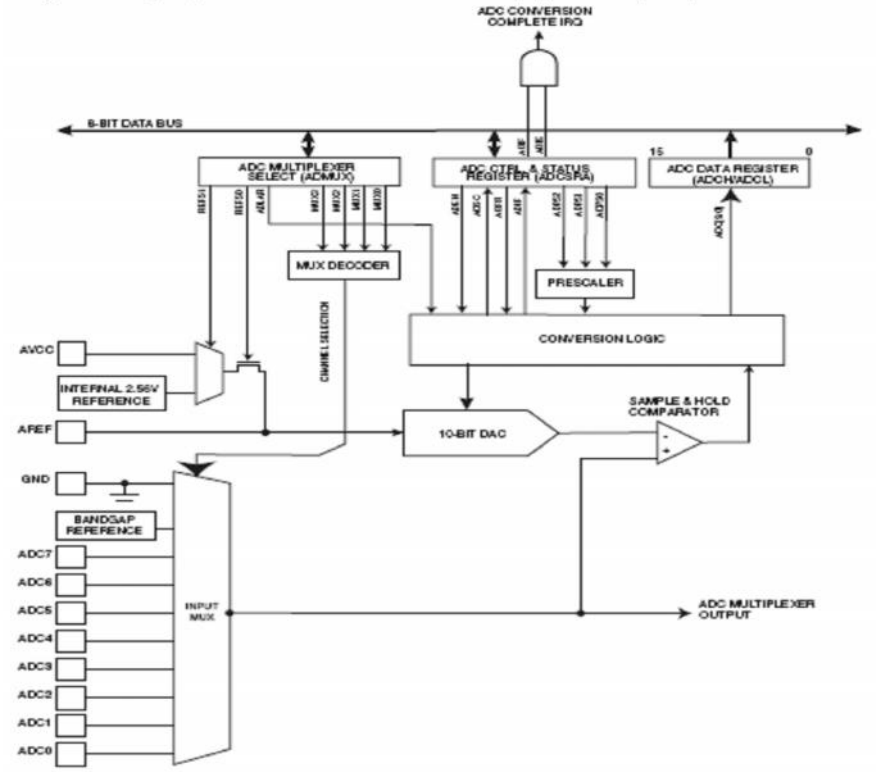
ATmega8 имеет 10-bit ADC последовательного приближения. ADC подключен к 8-ми канальному Аналоговому Мультиплексору который позволяет подключить вход ADC к любому из восьми входов (выводу Port C). Входы АЦП, при отсутствии входного напряжения, принимают значение 0V (GND).

ADC содержит блоки выборки и хранения, для запоминания входного напряжения на время преобразования. Функциональная схема ADC показана на рис. 90.

ADC имеет отдельный вывод питания, AVCC. Напряжение питания ADC (вывод AVCC) не должно отличаться от VCC более чем на ± 0.3V. Как подключить этот вывод, смотри в разделе "ADC компенсатор шума" на странице 201.

Встроенный источник опорного напряжения имеет номинальное напряжение 2.56V или AVCC, при условии, что внутренний ИОН «включен». ИОН может быть «стабилизирован» путем подключения к выводу AREF конденсатора (на GND), для уменьшения шума.

Рисунок 90.Функциональная схема аналогово-цифрового преобразователя



ADC преобразует входное аналоговое напряжение в 10-bit цифровое значение применяя последовательное приближение. Минимальное значение соответствует уровню (напряжению) GND, а максимальное соответствует напряжению на выводе AREF минус 1 LSB (бит младшего разряда). При необходимости, AVCC или внутренний ИОН (2.56V) могут быть подключены к AREF. Для этого надо записать биты REFSn в регистре ADMUX. Внутренний ИОН должен фильтроваться конденсатором на выводе AREF для уменьшения шума.

Подключение выбранного входного канала производится записью битов MUX в регистре ADMUX. Каждый вход ADC, измеряющий напряжение в диапазоне от GND до напряжения ИОН, должен быть определен как одиночный вход ADC. Это определяется битами ADEN в регистре ADCSRA. Выбор источника ИОН и входа ADC не будет иметь значения до тех пор, пока не установлен ADEN. ADC не подключен (не потребляет мощности) когда бит ADEN сброшен, этим способом рекомендуется выключать ADC перед переходом в «спящий режим».

ADC возвращает 10-bit результат, который располагается в ADC регистрах данных, ADCH и ADCL. По умолчанию, имеющийся результат выравнивается (сдвигается) в правую сторону **(прим.пер. для 10 битного преобразования)**, но результат может выравниваться и в левую сторону**(прим.пер. для 8 битного преобразования, два младших бита отбрасываются, теряются)**, если установлен бит ADLAR регистра ADMUX.

Если результат выравнивается влево и требуется не более 8-бит, то достаточно прочесть регистр ADCH. В других случаях, сначала должен быть прочитан регистр ADCL, а затем ADCH, пока не будет прочитан ADCH содержание Data Registers не изменится и будет соответствовать тому же самому преобразованию. Как только начинается чтение ADCL, доступ ADC к Data Registers блокируется. Это означает, что если преобразование ADC происходит после чтения ADCL, и завершается до прочтения ADCH, то ни какой регистр **(прим.пер. ADCL и ADCH)** не изменит свое значение и результат преобразования будет потерян. Когда чтение ADCH заканчивается, доступ ADC к регистрам ADCH и ADCL возобновляется.

ADC имеет собственное прерывание, которое может быть вызвано когда преобразование завершается. Когда доступ ADC к Data Registers запрещен между чтением ADCH и ADCL, прерывание будет вызвано даже если результат преобразования потерян.

Запуск преобразования

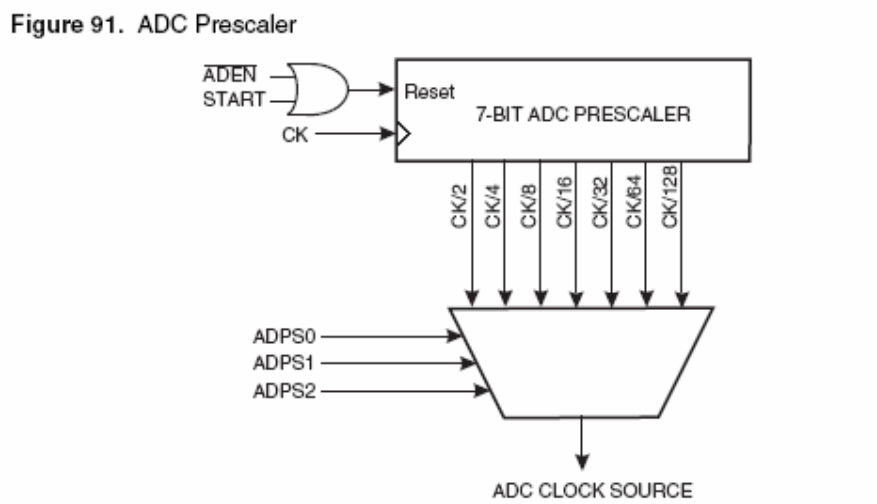
Одиночное преобразование начинается когда записывается логическая единица в ADC Start Conversion bit, ADSC. Этот бит остается в состоянии логической единицы в течение всего преобразования и аппаратно сбрасывается когда преобразование завершено. Если, во время преобразования, выбран другой входной канал, то ADC завершит текущее преобразование перед переключением на новый канал.

В режиме Непрерывной Работы ADC постоянно производит выборку и обновляет Data Register. Режим Непрерывной Работы выбирается установкой бита ADFR в ADCSRA. Запуск преобразования, в этом режиме, происходит после установки бита ADSC в ADCSRA. В этом режиме ADC производит преобразование (последовательное приближение) независимо от того, установлен ли флаг прерывания или нет, т.е. бит ADIF очищен или нет.

Предделитель и время преобразования

Рисунок 91. Предделитель АЦП

Figure 91. ADC Prescaler



По умолчанию, последовательное приближение производится на частотах от 50 кГц до 200 кГц, при максимальном разрешении. Если необходимо разрешение менее 10 бит, то частота преобразования может быть более 200 кГц, для получения более высокой скорости преобразования.

ADC содержит предделитель, который формирует допустимую частоту преобразования из любой частоты CPU, если она более 100 kHz. Предделитель устанавливается битами ADPS в ADCSRA. Предделитель начинает работать когда, установкой бита ADEN в ADCSRA, включается ADC. Предделитель работает все время пока установлен бит ADEN, и сбрасывается когда ADEN равен нулю.

Когда запущено одиночное (разовое) преобразование, путем установки бита ADSC в ADCSRA, преобразование начинается по первому положительному (из 0 в 1) фронту ADC clock cycle. Нормальное преобразование

длится 13 тактов частоты ADC clock cycles. Первое преобразование после включения ADC (установка бита ADEN в ADCSRA) требует 25 тактов ADC clock cycles для инициализации аналоговой части.

Фактическая выборка и запоминание входного сигнала происходит в течение 1.5 тактов ADC clock cycles после начала при обычном преобразовании и в течение 13.5 тактов ADC clock cycles после начала при первоначальном включении. Когда преобразование завершено, результат записывается в ADC Data Registers, и устанавливается бит ADIF. В режиме разового преобразования, бит ADSC сбрасывается одновременно с установкой ADIF. Программно бит ADSC может бить установлен снова, и новое преобразование начнется по первому положительному фронту ADC clock.

В режиме Free Running, новое преобразование начинается сразу же по завершении предыдущего преобразования, пока бит ADSC остается высоким. Время преобразования в различных режимах показано в Table 73.

Рисунок 92. Временная диаграмма работы АЦП при первом преобразовании в режиме одиночного преобразования

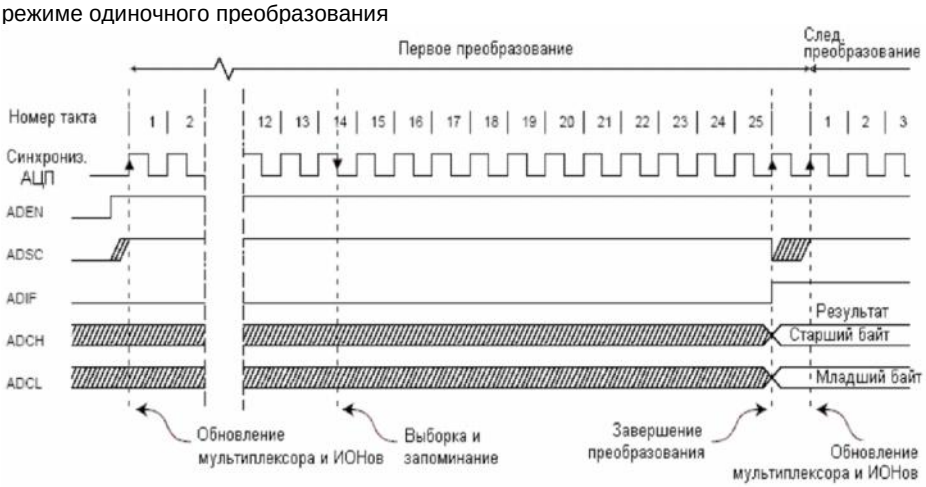


Рисунок 93. Временная диаграмма работы АЦП в режиме одиночного преобразования

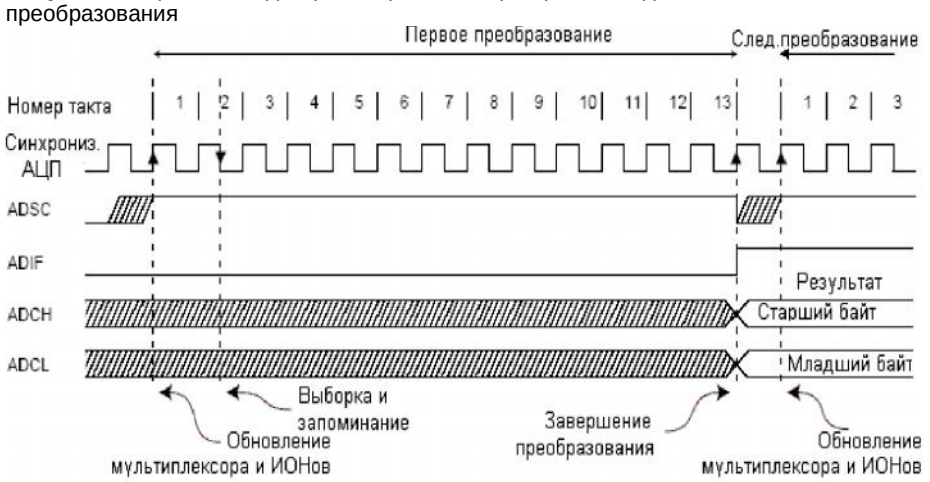


Рисунок 94. Временная диаграмма работы АЦП в режиме автоматического перезапуска(Free Running)

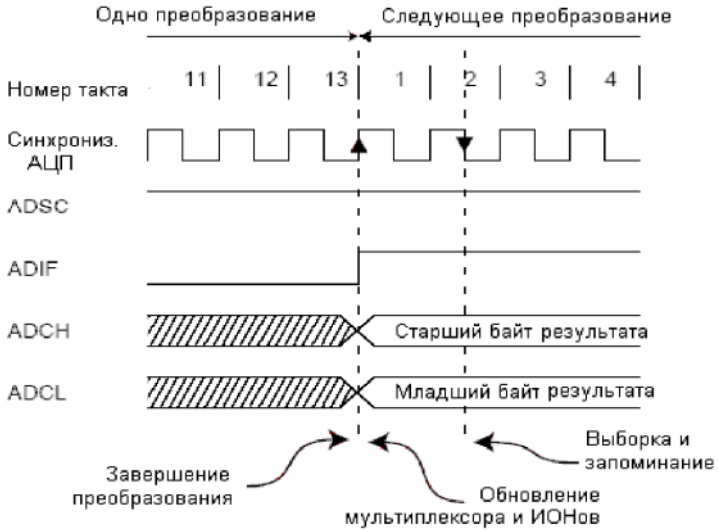


Таблица 73. Время преобразования АЦП

Тип преобразования	Длительность выборки-хранения (в тактах с момента начала преобразования)	Время преобразования (в тактах)
Первое преобразование	14.5	25
Нормальное однополярное преобразование	1.5	13

Выбор канала или источника опорного напряжения

Биты MUXn и REFS1:0 в регистре ADMUX являются единственными буферизированными через временный регистр, к которому CPU имеет произвольный доступ. Это гарантирует, что изменение входного канала и источника опорного напряжения будет происходить только в безопасный момент преобразования (по окончании одного и до начала следующего преобразования). Изменение канала и ИОН может быть проведено в любой момент до начала преобразования. После начала преобразований (с приходом первого импульса) выбор канала и ИОН запрещается на время всего преобразования. Продолжение выбора канала и ИОН начинается в последнем цикле ADC clock cycle перед завершением преобразования (когда бит ADIF в ADCSRA устанавливается). Учтите, что преобразование начинается по переднему (положительному) фронту ADC clock после того, как бит ADSC установлен. Тем самым, изменение канала и ИОН в регистре ADMUX не может быть произведено, пока не пройдет один импульс ADC clock после установки бита ADSC.

Если установлены биты ADFR и ADEN, то прерывание по окончанию преобразования может происходить в любое время. Если регистр ADMUX изменен во время обработки прерывания, то не возможно определить с какими настройками (прежними или новыми) будет произведено следующее преобразование. Содержание регистра ADMUX должно изменяться при следующих условиях:

1. Когда бит ADFR или ADEN сброшен.
2. В течение преобразования, по прошествии минимум одного цикла ADC clock после установки бита окончания преобразования.
3. После окончания преобразования, но перед тем как будет сброшен флаг прерывания (ИМХО до окончания обработки прерывания).

Если изменение содержимого регистра ADMUX произведено с соблюдением одного из этих условий, то следующее преобразование будет проведено с учетом новых установок.

Входные каналы ADC

При выборе входного канала необходимо следить за тем, что выполнены следующие условия правильного выбора канала:

При одиночном преобразовании выбор канала всегда производится перед началом преобразования. Выбор другого канала может быть произведен, не ранее чем, в первый цикл ADC clock после установки бита ADSC. Однако, самый простой метод – это ждать окончания преобразования, а затем выбирать (изменять) входной канал.

В режиме Непрерывной работы канал всегда выбирается до начала первого преобразования. Выбор другого канала может быть произведен, не ранее чем, в первый цикл ADC clock после установки бита ADSC. Однако, самый простой метод – это ждать окончания преобразования, а затем выбирать (изменять) входной канал. Так как следующее преобразование уже началось автоматически, то его результат отразит предыдущий выбор канала. Последующие преобразования отразят новый выбор канала.

Источник опорного напряжения ADC

Опорное (образцовое) напряжение ADC (VREF) определяет диапазон преобразования ADC. Если, измеряемое напряжение превышает VREF, то результатом одиночного преобразования будет значение 0x3FF. VREF может быть выбран как напряжение AVCC, встроенный ИОН 2.56V, или внешний источник, подключенный к выводу AREF.

AVCC соединен с ADC через пассивный переключатель. Встроенный ИОН 2.56V состоит из внутреннего стабилизатора (VBG) и встроенного усилителя. В любой микросхеме ATmega8, внешний вывод AREF подсоединен непосредственно к ADC, и стабильность внутреннего ИОН может быть увеличена путем подключения блокировочного конденсатора между выводом AREF и общим проводом. Напряжение VREF может быть измерено вольтметром в высоким входным сопротивлением на выводе AREF. Помните, что измерять VREF можно только вольтметром с высокоомным входом, и только при подключенном блокировочном конденсаторе.

Если используется внешний источник опорного напряжения, который подсоединен к выводу AREF, то использовать внутренние ИОН (AVCC или 2.56V) невозможно, так как они будут зашунтированы внешним напряжением. Если внешний ИОН не подключен к выводу AREF, то в качестве ИОН используется или AVCC или 2.56V. Первое, после выбора нового ИОН, преобразование может быть неточным, поэтому его следует не учитывать (пропускать).

ADC шумоподавитель

ADC имеет шумоподавитель, который обрабатывает преобразование в спящем режиме, для устранения помех, создаваемых CPU и периферийными устройствами. Шумоподавитель может использоваться и в режиме Idle (пониженного энергопотребления). Для использования шумоподавителя необходимо выполнить следующее:

1. ADC должно бить разрешено и не занято преобразованием. Должен быть выбран режим одиночного преобразования и разрешено прерывание по окончанию преобразования.
2. Установите режим Уменьшения Шума ADC (или режим Idle). ADC начнет преобразование только когда будет остановлен CPU.
3. Если никакие другие прерывания не произойдут до момента завершения преобразования ADC, то прерывание от ADC «разбудит» CPU, который перейдет на обработку этого прерывания. Если другое прерывание «пробуждает» CPU до завершения преобразования ADC, то будет выполнено это (разбудившее) прерывание, и запрос на прерывание от ADC будет сгенерирован по окончании преобразования. CPU останется в активном режиме пока не получит новую команду на остановку (sleep command is executed).

Помните, что ADC не выключится автоматически при новой остановке CPU в режиме Idle и режиме Уменьшения Шума. Для остановки ADC необходимо сбросить бит ADEN перед переходом в спящий режим, для того чтобы избежать потребления мощности ADC.

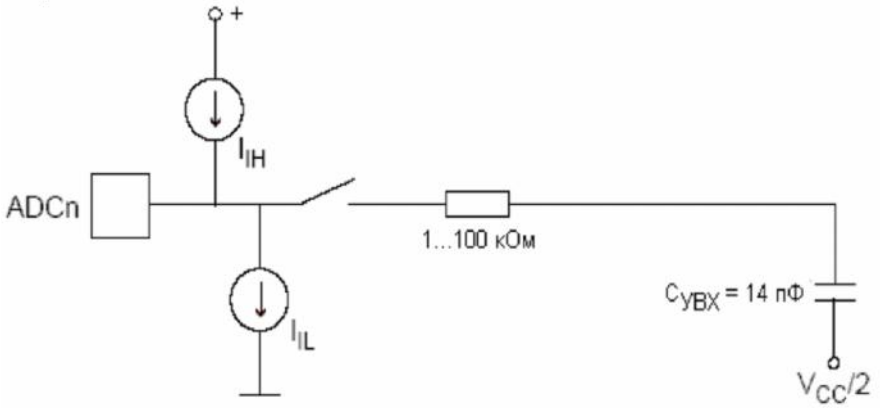
Схема Аналогового Входа

Схема аналогового входа для одиночного канала показана на рис. 95. Каждый аналоговый вход ADCn, имеет внутренний резистор и конденсатор, во всех каналах ADC. Когда канал подключен, к источнику сигнала подключается резистор и конденсатор S/H (комбинированное сопротивление).

ADC оптимизирован для аналоговых сигналов с полным выходным сопротивлением не более 10 кОм. При этом время выборки (преобразования) будет небольшим. Если источник обладает более высоким сопротивлением, то время преобразования будет определятся временем заряда (от источника) конденсатора S/H, и может меняться в широких пределах. Рекомендуется использовать источники сигнала с низким выходным сопротивлением и медленно изменяющимися сигналами, так как это минимизирует время перезаряда конденсатора S/H.

Изменение входного сигнала с частотой выше чем (fADC/2) недопустимо, так как может привести к непредсказуемым искажениям при преобразовании. Рекомендуется удалять высокочастотные составляющие сигнала путем установки ФНЧ на входе ADC.

Рисунок 95. Схема аналогового входа

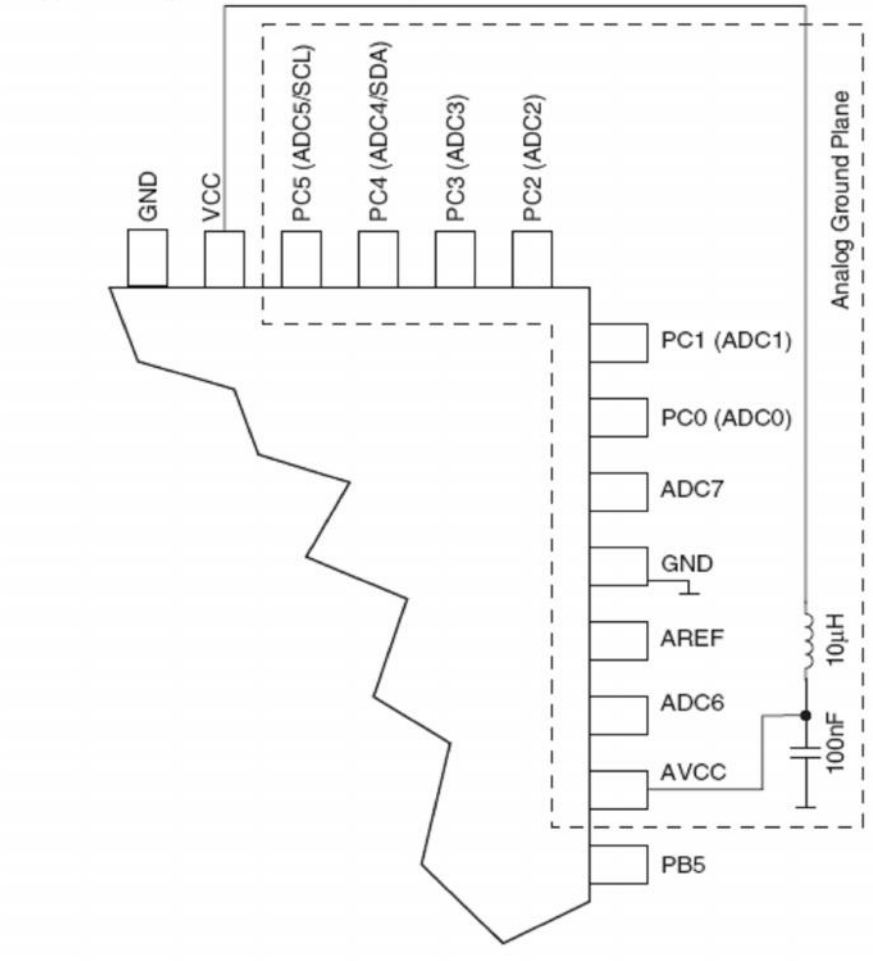


Методы(способы) Шумоподавления

Цифровая часть устройства как внутри, так и снаружи генерирует EMI (электро - магнитные импульсы), которые могут воздействовать на точность аналоговых значений. Если точность преобразования аналогового сигнала критична, то уровень шума может быть снижен следующими методами:

1. Проводники (дорожки на плате) аналогового сигнала должны быть максимально короткими.
2. Вывод AVCC должен быть соединен с выводом VCC через LC фильтр, как показано на рис. 96.
3. Используйте функцию Шумоподавления для уменьшения помех от CPU.
4. Если любой из выводов портов ADC [3..0] используется как цифровой выход, то необходимо чтобы он (они) не переключались во время преобразования входного аналогового сигнала. Однако, использование Two-wire Interface (I2C) (ADC4 и ADC5) будет воздействовать только на преобразование на выводах ADC4 и ADC5 и не влияет на другие каналы ADC.

Рисунок 96. Подключение питания АЦП



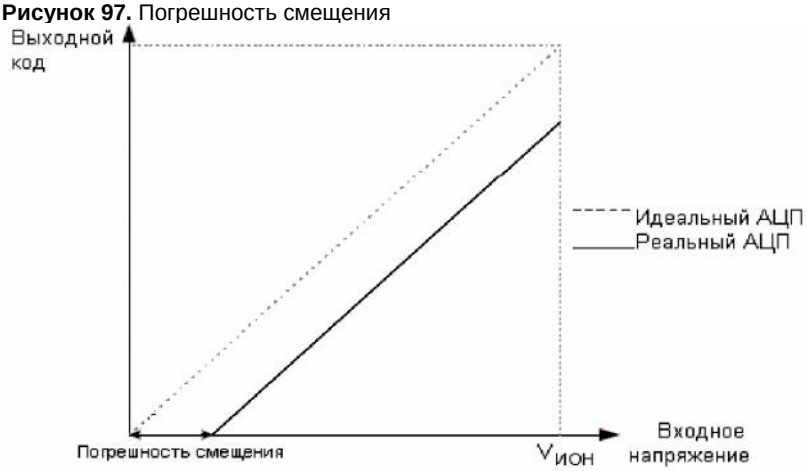
Точность преобразования ADC

Одиночное, n-битное преобразование ADC переводит величину входного напряжения в диапазоне от GND до VREF линейно за 2n шагов (LSBs). Самый низкий код читается как 0, а самый высокий как 2n-1.

Отдельные параметры описывают отклонение преобразователя от идеального:

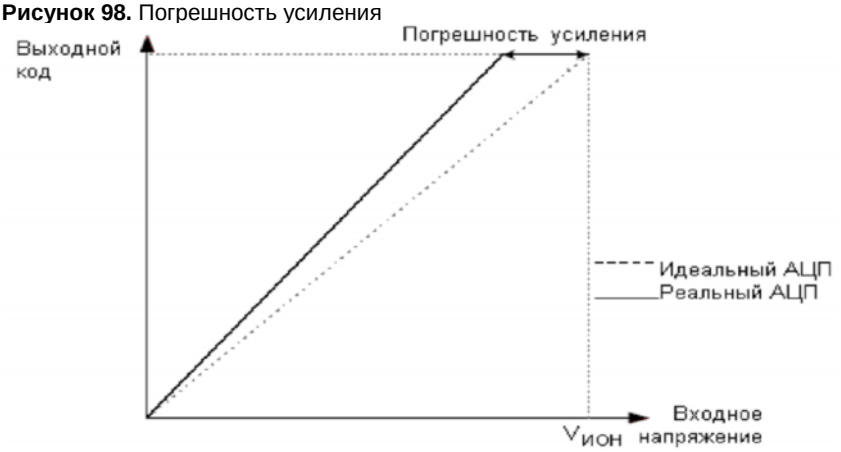
- Начальное Смещение: Отклонение первого (начального) перемещения (итерации) (0x000 to 0x001) сравниваемое с идеальным перемещением (0.5 LSB). Идеальное значение: 0 LSB.

Рисунок 97. Погрешность смещения



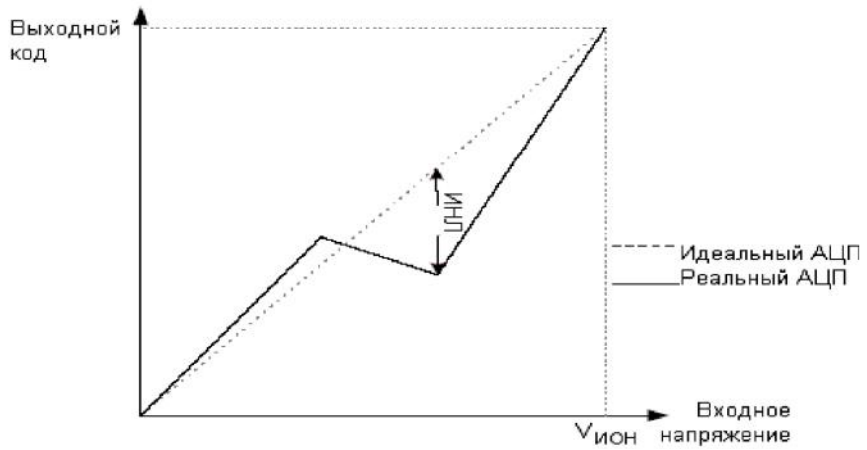
• Ошибка Усиления: После корректировки Начального Смещения, Ошибка Усиления определяется как отклонение последнего перемещения (итерации) (0x3FE к 0x3FF) от идеального перемещения (в 1.5 LSB ниже максимума). Идеальная величина: 0 LSB

Рисунок 98. Погрешность усиления



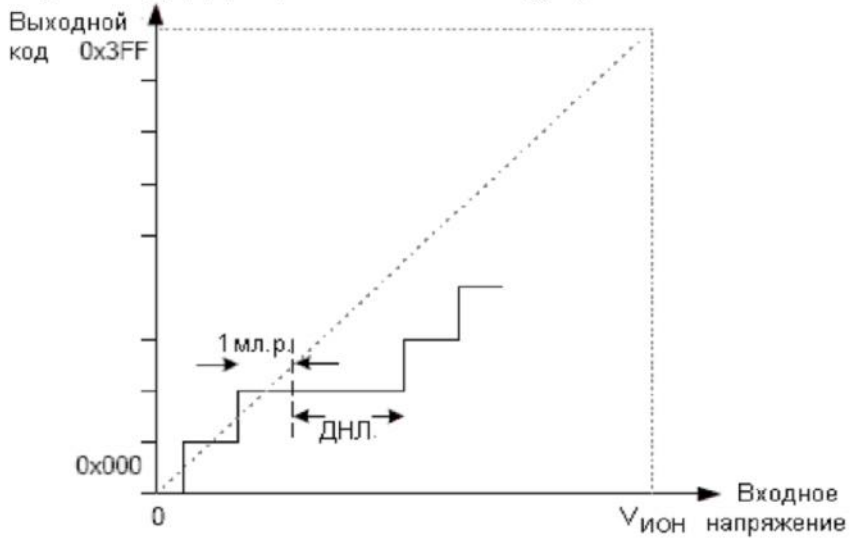
• Интегральная Нелинейность(INL): После корректировки Начального Смещения и Ошибки Усиления, INL это максимальное отклонение фактического перемещения, сравниваемого с идеальным перемещением для любого участка. Идеальная величина: 0 LSB.

Рисунок 99. Интегральная нелинейность (ИНЛ)



• Дифференциальная Нелинейность (DNL): Максимальное отклонение фактической ширины кода (в интервале между двумя смежными приращениями) от идеальной ширины кода (1 LSB). Идеальная величина: 0 LSB.

Рисунок 100. Дифференциальная нелинейность (ДНЛ)



• Ошибка квантования: Из-за того, что входное напряжение преобразуется в конечное число кодов, минимальное и максимальное значение входного напряжения , в диапазоне единицы квантования (1 LSB), будет иметь одно и то же значение. Ошибка квантования всегда равна ±0.5 LSB.

• Абсолютная точность: Абсолютная точность – это максимальное отклонение фактического (нескорректированного) приращения (итерации), которое сравнивается с идеальным перемещением. Это обобщенная ошибка Начального Смещения, Ошибки Усиления, Интегральной Нелинейности, Дифференциальной Нелинейности, Ошибки квантования. Идеальная величина: ±0.5 LSB.

Результат Преобразования ADC

После окончания преобразования (установлен бит ADIF), результат этого преобразования находится в ADC Result Registers (ADCL, ADCH).
Для единичного преобразования результат определяется как –

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

где VIN – напряжение на выбранном выводе и VREF выбранное опорное напряжение (смотри Table 74 на странице 203 и Table 75 на странице 203). 0x000 соответствует напряжению общего провода (GND), и 0x3FF соответствует напряжению ИОН минус один разряд LSB.

Переключение Мультиплексора ADC
Регистр - ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

· Биты 7:6 - REFS1:0: Биты выбора ИОН

Этими битами выбирается ИОН для ADC в соответствие с Табл. 74. Если эти биты изменены пока идет преобразование, то изменение вступит в силу после окончания текущего преобразования (установиться бит ADIF в ADCSRA). Внутренний (встроенный) ИОН не может быть использован если к выводу AREF подключен внешний ИОН.

Таблица 74. Выбор опорного источника АЦП

REFS1	REFS0	Опорный источник
0	0	AREF, внутренний ИОН отключен
0	1	AVCC с внешним конденсатором на выводе AREF
1	0	Зарезервировано
1	1	Внутренний источник опорного напряжения 2.56В с внешним конденсатором на выводе AREF

· Bit 5 - ADLAR: Выравнивание результата в левую сторону

Бит ADLAR воздействует на результат преобразования, находящийся в ADC Data Register. Установите бит ADLAR для левого выравнивания результата. Иначе, результат выравнивается вправо. Изменение бита ADLAR будет воздействовать на ADC Data Register немедленно, независимо от любых продолжающихся преобразований. Полное описание действия этого бита приводиться в разделе "The ADC Data Register - ADCL and ADCH" на странице 208.

· Bits 3:0 - MUX3:0: Переключение Аналоговых Каналов (Входов)

Установкой этих битов выбирается вход, подключаемый к ADC. Для выяснения деталей смотри Table 75. Если эти биты изменены в течение преобразования, то изменение не будет иметь силы пока текущее преобразование не завершится (устанавливается бит ADIF в ADCSRA).

Таблица 75. Выбор входного канала

MUX3..0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	
1001	
1010	
1011	
1100	
1101	
1110	1.23V (V_{BG})
1111	0V (GND)

ADC Управление и Состояние
Регистр A - ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

· Бит 7 - ADEN: ADC Включение
Установка этого бита включает ADC. А сброс этого бита выключает ADC. Переключение ADC во время преобразования приведет к завершению этого преобразования.

· Бит 6 - ADSC: ADC Старт Преобразования
В режиме одиночных преобразований установка этого бита приведет к началу следующего (каждого) преобразования. В режиме непрерывного преобразования установка этого бита приведет к старту первого преобразования. Первое (начальное) преобразование после установки бита ADSC которая (установка) произошла после включения ADC, или если бит ADSC устанавливается одновременно с включением ADC, займет (преобразование) 25 циклов ADC clock вместо 13 в нормальном режиме. Это первое (начальное) преобразование выполняет инициализацию ADC.
ADSC будет читаться как единица пока преобразование не завершено. При завершении преобразования этот бит сбрасывается аппаратно. Запись нуля в этот бит не будет иметь эффекта.

· Бит 5 - ADFR: ADC Выбор режима Непрерывного Преобразования
Когда установлен этот бит, то ADC переходит в режим Непрерывного Преобразования. В этом режиме преобразование и запись результата в Data Registers происходит постоянно. Сброс этого бита выключает режим Непрерывного Преобразования.

· Бит 4 - ADIF: ADC Установка прерывания
Этот бит устанавливается когда ADC преобразование завершении и результат записан в Data Registers. Прерывание ADC обрабатывается если биты ADIE и I-бит в регистре SREG установлены. Бит ADIF очищается аппаратно когда заканчивается обработка прерывания по вектору прерывания. Можно сбросить этот бит, записав в него (to the flag) единицу. Будьте внимательны при проведении цикла Чтение-Модификация-Запись регистра ADCSRA, так как может быть сброшено отложенное прерывание. Это(сброс отложенного прерывания) так же актуально при использовании команд SBI CBI.

· Бит 3 - ADIE: ADC Разрешение прерывания
Когда этот бит установлен, и установлен бит I регистра SREG, прерывание по завершению преобразования ADC разрешено.

· Биты 2:0 - ADPS2:0: ADC Выбор частоты предделетеля

Эти биты определяют соотношение между тактовой частой процессора и частотой дискретизации ADC.

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

Регистры данных ADC - ADCL and ADCH

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Когда преобразование завершено, результат находится в этих двух регистрах.
Когда читается регистр ADCL, ADC Data Register не будет изменятся пока не будет прочитан ADCH. Следовательно, если результат выравнивается по левому краю и требуется 8-ми разрядная точность, то достаточно читать ADCH. Иначе сначала должен читаться ADCL, а затем ADCH.
Бит ADLAR в регистре ADMUX, и биты MUXn в регистре ADMUX определяют способ, которым считывается результат. Если ADLAR установлен, то результат выровнен влево. Если бит ADLAR сброшен (по умолчанию), то результат выровнен по правому краю.

· Биты ADC9:0: ADC Результат преобразования

В этих битах находится результат преобразования, детально они рассмотрены в главе "ADC Результат преобразования" на странице 205.

Программирование памяти

Биты защиты памяти программ и данных

ATmega8 содержит 6 битов защиты, которые можно оставить в незапрограммированном состоянии ("1") или же запрограммировать ("0") для активизации дополнительных функций, представленных в таблице 86. Стирание бит защиты (установка "1") может быть выполнена только командой стирание кристалла (Chip Erase).

Таблица 86. Байт с битами защиты

Биты защиты	Разряд	Описание	Исходное значение
	7	-	1 (незапрограммированный)
	6	-	1 (незапрограммированный)
BLB12	5	Бит защиты загрузочного сектора	1 (незапрограммированный)
BLB11	4	Бит защиты загрузочного сектора	1 (незапрограммированный)
BLB02	3	Бит защиты загрузочного сектора	1 (незапрограммированный)
BLB01	2	Бит защиты загрузочного сектора	1 (незапрограммированный)
LB2	1	Бит защиты	1 (незапрограммированный)
LB1	0	Бит защиты	1 (незапрограммированный)

Прим.: "1" означает незапрограммированное состояние, а "0" - запрограммированное.

Таблица 117. Режимы защиты

Биты защиты памяти			Тип защиты
Режим LB	LB2	LB1	
1	1	1	Нет защиты памяти.
2	1	0	Дальнейшее программирование флэш-памяти и ЭСППЗУ отключено при параллельном и последовательном (SPI/JTAG) программировании. Конфигурационные биты защищены при любом способе программирования[1]
3	0	0	Дальнейшее программирование и проверка флэш-памяти и ЭССПЗУ отключена как при параллельном, так и при последовательном программировании через SPI/JTAG. Конфигурационные биты защищены при любом способе программирования[1]
Режим BLB0	BLB02	BLB01	
1	1	1	Нет ограничений действия инструкций SPM или (E)LPM при адресации сектора прикладной программы.
2	1	0	SPM не записывает данные в сектор прикладной программы.
3	0	0	SPM не записывает данные в сектор прикладной программы, а выполнение инструкции (E)LPM в загрузочном секторе не позволяет считать данные из сектора прикладной программы. Если векторы прерываний размещены в загрузочном секторе, то при выполнении команд в секторе прикладной программы прерывания отключаются.
4	0	1	Выполнение (E)LPM в загрузочном секторе не позволяет считать данные из сектора прикладной программы. Если векторы прерываний размещены в загрузочном секторе, то при выполнении команд в секторе прикладной программы прерывания отключаются.
Режим BLB1	BLB12	BLB11	

1	1	1	Нет ограничений действия инструкций SPM или (E)LPM при адресации загрузочного сектора.
2	1	0	SPM не записывает данные в загрузочный сектор.
3	0	0	SPM не записывает данные в загрузочный сектор, а выполнение инструкции (E)LPM в секторе прикладной программы не позволяет считать данные из загрузочного сектора. Если векторы прерываний размещены в секторе прикладной программы, то при выполнении команд в загрузочном секторе прерывания отключаются.
4	0	1	Выполнение (E)LPM в секторе прикладной программы не позволяет считать данные из загрузочного сектора. Если векторы прерываний размещены в секторе прикладной программы, то при выполнении команд в загрузочном секторе прерывания отключаются.

Прим.:
1. Конфигурационные биты необходимо программировать перед программированием бит защиты.
2. "1" означает незапрограммированное состояние, а "0" - запрограммированное.

Конфигурационные биты

ATmega8 имеет два конфигурационных байта. Таблицы 87 - 88 кратко описывают функционирование и расположение всех конфигурационных бит. Обратите внимание, что если конфигурационный бит запрограммирован, то при его считывании возвращается лог. 0.

Таблица 87. Старший конфигурационный байт

Наименование бита	Разряд	Описание	Исходное значение
RSTDISBL(4)	7	Выбор, PC6 - контакт ввода-вывода или контакт СБРОСА	1 (незапрограммированный, PC6 - КОНТАКТ СБРОСА),
WDTON	6	WDT всегда вкл.	1 (незапрограммированный, WDT, разрешено WDTCR)
SPIEN(1)	5	Разрешение последовательной загрузки программы и данных	0 (запрограммированное, программирование через SPI разрешено)
CKOPT(2)	4	Настройка генератора	1 (незапрограммированное)
EESAVE	3	Запрет стирания EEPROM командой стирание кристалла	1 (незапрограммированное, стирание кристалла вызывает стирание EEPROM)
BOOTSZ1	2	Выбор размера загрузочного сектора (см. табл. 82)	0 (запрограммированное) (3)
BOOTSZ0	1	Выбор размера загрузочного сектора (см. табл. 82)	0 (запрограммированное) (3)
BOOTRST	0	Выбор вектора сброса	1 (незапрограммированное)

Прим.:
1. Конфигурационный бит SPIEN недоступен в режиме последовательного программирования через SPI.
2. Функционирование конфигурационного бита CKOPT зависит от установок бит CKSEL. См. "Источники синхронизации".
3. Исходное значение бит BOOTSZ1..0 соответствует выбору максимальному размеру загрузочного сектора. См. таблицу 82.
4. Когда RSTDISBL запрограммирован используется режим Параллельного программирования, чтобы изменить предохранители или выполнить далее программирование.

Таблица 87. Младший конфигурационный байт

Наименование бит	Разряд	Описание	Исходное значение
BODLEVEL	7	Порог срабатывания супервизора питания	1 (незапрограммированное)
BODEN	6	Разрешение супервизора питания	1 (незапрограммированное, супервизор отключен)
SUT1	5	Выбор времени запуска	1 (незапрограммированное)(1)
SUT0	4	Выбор времени запуска	0 (запрограммированное)(1)
CKSEL3	3	Выбор тактового источника	0 (запрограммированное)(2)
CKSEL2	2	Выбор тактового источника	0 (запрограммированное)(2)
CKSEL1	1	Выбор тактового источника	0 (запрограммированное)(2)
CKSEL0	0	Выбор тактового источника	1 (незапрограммированное)(2)

Прим.:
1. Исходные установки SUT1..0 соответствуют выбору максимальному времени старта. Подробности представлены в таблице 10.
2. Исходные установки CKSEL3..0 соответствуют выбору внутреннего RC-генератора частотой 1 МГц. Подробности представлены в таблице 2.

На состояние конфигурационных бит не оказывает влияния команда стирания кристалла "Chip Erase". Обратите внимание, что доступ к конфигурационным битам заблокирован, если запрограммирован бит защиты LB1. Поэтому, конфигурационные биты необходимо программировать перед программированием бит защиты.

Защита конфигурационных бит

Доступ к конфигурационным битам блокируется, если микроконтроллер перешел в режим программирования и изменение их значений не даст никакого эффекта до тех пор, пока микроконтроллер не выйдет из режима программирования. Данное не распространяется на бит EESAVE и он может быть запрограммирован в любой момент. Доступ к конфигурационным битам также блокируется при подаче питания в нормальном режиме работы (не программировании).

Сигнатурные байты

Все микроконтроллеры Atmel имеют трехбайтный сигнатурный код, который позволяет идентифицировать устройство. Код можно считать как в параллельном, так и в последовательном режимах программирования, в т.ч. когда микроконтроллер защищен. У каждого байта сигнатурного кода имеется свой собственный адрес и назначение.

Для ATmega8 сигнатурными байтами являются:
1. $000: $1E (идентифицирует производителя: Atmel)
2. $001: $93 (идентифицирует размер флэш-памяти: 8 кбайт)
3. $002: $07 (идентифицирует тип микроконтроллера: ATmega8)

Калибровочный байт

Внутри ATmega8 хранятся четыре различных калибровочных значений для внутреннего RC- генератора. Данные значения хранятся по адресам 0x000, 0x0001, 0x0002 и 0x0003 и соответствуют частотам генератора 1, 2, 4 и 8 МГц. В процессе сброса в регистр OSCCAL автоматически записывается значение калибровочного байта для частоты 1 МГц. Если используется другая частота, то соответствующее значение должно быть вручную записано в регистр OSCCAL (см. "Регистр калибровки генератора - OSCCAL"). на странице 31

Размер страницы

Таблица 89. Количество слов в странице и количество страниц во флэш-памяти

Размер флэш-памяти	Размер страницы	**PCWORD**	Кол. страниц	**PCPAGE**	**PCMSB**
4 кслов (8кбайт)	32 слова	PC[4:0]	128	PC[11:5]	11

Таблица 90. Количество слов в странице и количество страниц в ЭСППЗУ

Размер EEPROM	Размер страницы	**PCWORD**	Кол. страниц	**PCPAGE**	**EEAMSB**
512 байт	4 байт	EEA[1:0]	128	EEA[8:2]	8

Параметры параллельного программирования, расположение выводов и команды

В данном разделе описывается как у ATmega8 запрограммировать и проверить флэш-память, EEPROM, биты защиты и конфигурационные биты. Полагается, что длительность импульсов не менее 250 нс, если не имеется других указаний

Наименование сигналов

На рисунке 104 и в таблице 91 показывается расположение и назначение выводов ATmega8, которые используются для параллельного программирования. Выводы XA1/XA0 определяют выполняемое действие при положительном фронте на выводе XTAL1 (см. табл. 93).

Подачей импульсов на WR или OE в зависимости от загруженной команды определяется выполняемое действие. Описание команд представлено в таблице 94.

Рисунок 104. Параллельное программирование

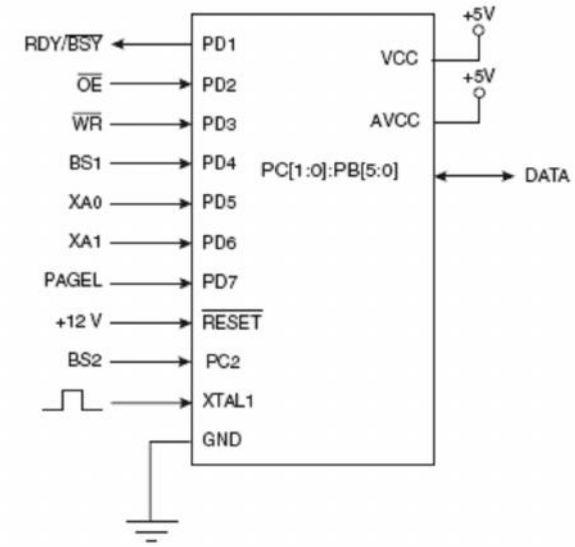


Таблица 91. Расположение и назначение выводов программирования

Наименование сигнала в режиме программирования	Наименование вывода	Направление	Функция
RDY/BSY	PD1	Выход	0 - микроконтроллер занят программированием;1 - микроконтроллер готов к загрузке новой команды
OE	PD2	Вход	Разрешение вывода (активный низкий)
WR	PD3	Вход	Строб записи (активный низкий)
BS1	PD4	Вход	Выбор байта 1 ("0" выбирает младший байт, "1" выбирает старший байт)
XA0	PD5	Вход	Код функции XTAL, разряд 0
XA1	PD6	Вход	Код функции XTAL, разряд 1
PAGEL	PD7	Вход	Загрузка страницы данных в память программ и ЭСППЗУ
BS2	PC2	Вход	Выбор байта 2 ("0" выбирает младший байт, "1" выбирает 2-ой старший байт)
DATA	{PC[1:0]: PB[5:0]}	Вход/ выход	Двунаправленная шина данных (выводит данные, когда OE=0)

Таблица 92. Состояния выводов, которые используются для входа в режим Программирования

Вывод	Обозначение	Значение
PAGEL	Prog_enable[3]	0
XA1	Prog_enable[2]	0
XA0	Prog_enable[1]	0
BS1	Prog_enable[0]	0

Таблица 93. Назначение кодов XA1 и XA0

XA1	XA0	Выполняемая функция во время импульса на XTAL1
0	0	Загрузка адреса флэш-памяти или ЭСППЗУ (какой байт адреса, старший или младший определяется BS1)
0	1	Загрузка данных (какой байт данных, старший или младший определяется BS1)
1	0	Загрузка команды
1	1	Нет никаких действий (холостой ход)

Таблица 94. Коды команд

Код команды	Функция
1000 0000	Стирание кристалла (Chip Erase)
0100 0000	Запись конфигурационных бит
0010 0000	Запись бит защиты
0001 0000	Запись флэш-памяти
0001 0001	Запись ЭСППЗУ
0000 1000	Чтение сигнатурных байт и калибровочного байта
0000 0100	Чтение конфигурационных бит и бит защиты
0000 0010	Чтение флэш-памяти
0000 0011	Чтение ЭСППЗУ

Параллельное программирование

Ввод режима программирования

Для ввода параллельного режима программирования необходимо выполнить действия в следующей последовательности:

1. Подать напряжение 4.5 - 5.5В между VCC и GND, задержка не менее 100 мкс.
2. Подаем лог. 0 на вход сброса "RESET" и переключаем вход XTAL1 не менее ШЕСТИ раз.
3. Устанавливаем на входах "Prog_enable" (см. табл. 92) код "0000" и ожидаем 100 нс.
4. Подаем напряжение 11.5 - 12.5В на "RESET". Если в течение 100 нс после подачи +12В на RESET произойдет изменения состояния входов Prog_enable, то это вызовет сбой режима программирования.

Обратите внимание, если контакт RESETотключается, программируя предохранитель RSTDISBL или используется внешний кварцевый резонатор или внешняя RC-цепь,то нет возможности приложить импульсы к XTAL1. В этом случае придерживаются следующей последовательности:

1. Установить код "0000" на входах Prog_enable.
2. Подать напряжение 4.5 - 5.5В между VCC и GND одновременно с подачей напряжения 11.5 - 12.5В на RESET.
<>Ожидаем 100 мкс.
3. Перепрограммируем конфигурационные биты для выбора в качестве источника синхронизации внешнего генератора (CKSEL3:0 = 0b0000). Если запрограммированы биты защиты, то предварительно необходимо выполнить команду стирания кристалла (Chip Erase).
4. Выходим из режима программирования выключением питания или путем подачи лог. 0 на RESET.
5. Ввод режима программирования, описанного выше.

Рекомендации по повышению эффективности программирования

Загружаемые команды и адрес запоминаются в микроконтроллере в процессе программирования. Для эффективности программирования необходимо учитывать следующее:

- Если выполняется чтение или запись по нескольким адресам памяти, то команда может быть загружена однократно.
- Если записываемое значение равно $FF, то его запись можно пропустить, т.к. после выполнения команды стирания кристалла все ячейки EEPROM (если конфигурационный бит EESAVE запрограммирован) и флэш-памяти заполнены этим кодом.
- Старший байт адреса необходимо загружать только перед началом программирования или чтения новых 256 слов флэш-памяти и 256 байт EEPROM. Данное распространяется также на чтение сигнатурных байт.

Стирание кристалла

Выполнение команды стирания кристалла приводит к очистке содержимого флэш-памяти и EEPROM (1), а также бит защиты. Очистка бит защиты происходит только после полного стирания памяти программ. Конфигурационные биты при этом не изменяются. Стирание кристалла необходимо выполнять перед перепрограммированием флэш-памяти и/или EEPROM .
Прим.:
1. Память EEPROM не изменяется, если конфигурационный бит EESAVE запрограммирован.
Ввод команды "Стирание кристалла":
1. Установка на XA1, XA0 кода "10". Этим разрешается команда загрузки.
2. Установка BS1 ="0".
3. Установка данных "1000 0000". Это команда "Стирание кристалла".
4. Формируем положительный фронт на XTAL1. Этим загружается команда.
5. Формируем отрицательный фронт WR. Этим запускаем механизм стирания кристалла. RDY/BSY переходит в низкое состояние.
6. Ожидаем, когда RDY/BSY перейдет в единичное состояние, а затем загружаем новую команду.

Программирование флэш-памяти

Флэш-память имеет постраничную организацию (см. табл. 89). Во время программирования флэш-памяти данные помещаются в страничный буфер. Это позволяет за один подход записать всю страницу. Ниже описана процедура программирования всей флэш-памяти:

A. Загрузка команды "Запись флэш-памяти "
1. Установка XA1, XA0 = "10". Этим разрешается загрузка команды.
2. Установка BS1 = "0".
3. Установка ДАННЫХ = "0001 0000". Это команда записи флэш-памяти.
4. Формируем положительный фронт на XTAL1. Этим загружается команда.

B. Загрузка младшего адресного байта
1. Установка XA1, XA0 = "00". Это разрешает загрузку адреса.
2. Установка BS1 = "0". Этим выбирается младший адрес.
3. Установка ДАННЫХ = мл. байт адреса ($00 - $FF).
4. Формируем положительный фронт на XTAL1. Этим загружается младший байт адреса.

C. Загрузка младшего байта адреса.
1. Установка XA1, XA0="01". Это разрешает загрузку данных.
2. Установка ДАННЫХ = мл. байт адреса ($00 - $FF).
3. Формируем положительный фронт на XTAL1. Этим загружается байт данных.

D. Загрузка старшего байта данных
1. Установка BS1 = "1". Этим выбирается старший байт данных.
2. Установка XA1, XA0 = "01". Этим выбирается загрузка данных.
3. Установка ДАННЫХ = ст. байт данных ($00 - $FF).
4. Формируем положительный фронт на XTAL1. Этим загружается байт данных.

E. Загрузка данных.
1. Установка BS1 = "1". Этим выбирается старший байт данных.
2. Формируем положительный фронт на PAGEL. Этим защелкиваются байты данных (см. форму сигналов на рисунке 106).

F. Повторяем B-E до заполнения всего буфера или до завершения загрузки данных в пределах страницы. Если младшие разряды адреса адресуют слова в пределах страницы, то старшие разряды адреса адресуют страницы в пределах флэш-памяти. Это иллюстрируется на рисунке 105. Обратите внимание, что если требуется менее 8-разрядов для адресации слов в странице (размер страницы < 256), то старшие разряды младшего адресного байта адресуют к странице при выполнении страничной записи.

G. Загрузка старшего адресного байта.
1. Установка XA1, XA0 = "00". Этим разрешается загрузка адреса.
2. Установка BS1 = "1". Этим выбирается старший байт адреса.
3. Установка ДАННЫХ = ст. адресному байту ($00 - $FF).
4. Формируем положительный фронт на XTAL1. Этим загружаем ст. адресный байт.

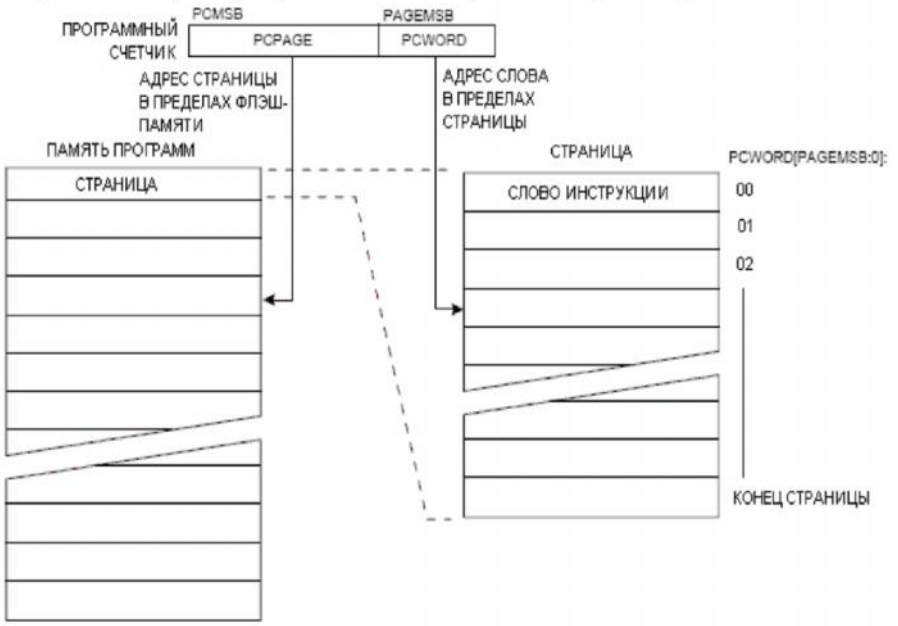
H. Программируем страницу

1. Установка BS1 = "0".
2. Формируем отрицательный фронт на WR. Этим инициируем программирование всей страницы данных. RDY/BSY переходит в низкое состояние.
3. Ожидаем появление лог. 1 на RDY/BSY (см. осциллограммы сигналов на рисунке 137).

I. Повторяем B-H до завершения программирования всей флэш-памяти.

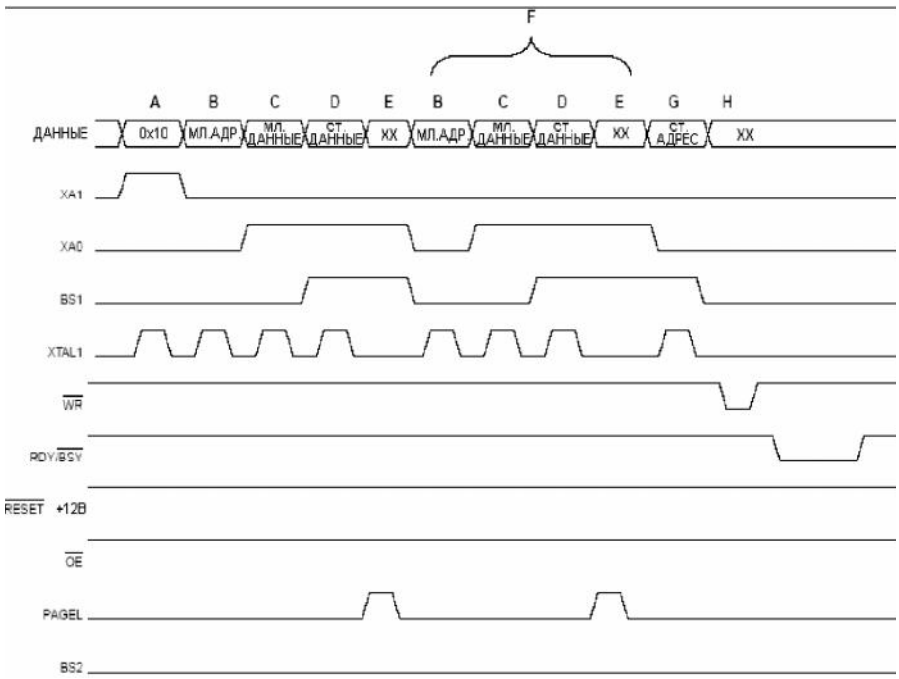
J. Завершения программирования страницы.
1. Установка XA1, XA0 = "10". Этим разрешается загрузка команды.
2. Установка ДАННЫХ = "0000 0000". Это команда "нет операции".
3. Формируем положительный фронт на XTAL1. Этим загружаем команду и сбрасываем внутренние сигналы записи.

Рисунок 105. Адресация флэш-памяти со страничной организацией



Прим.:
1. PCPAGE и PCWORD представлены в таблице 89.

Рисунок 106. Осциллограммы сигналов программирования флэш-памяти



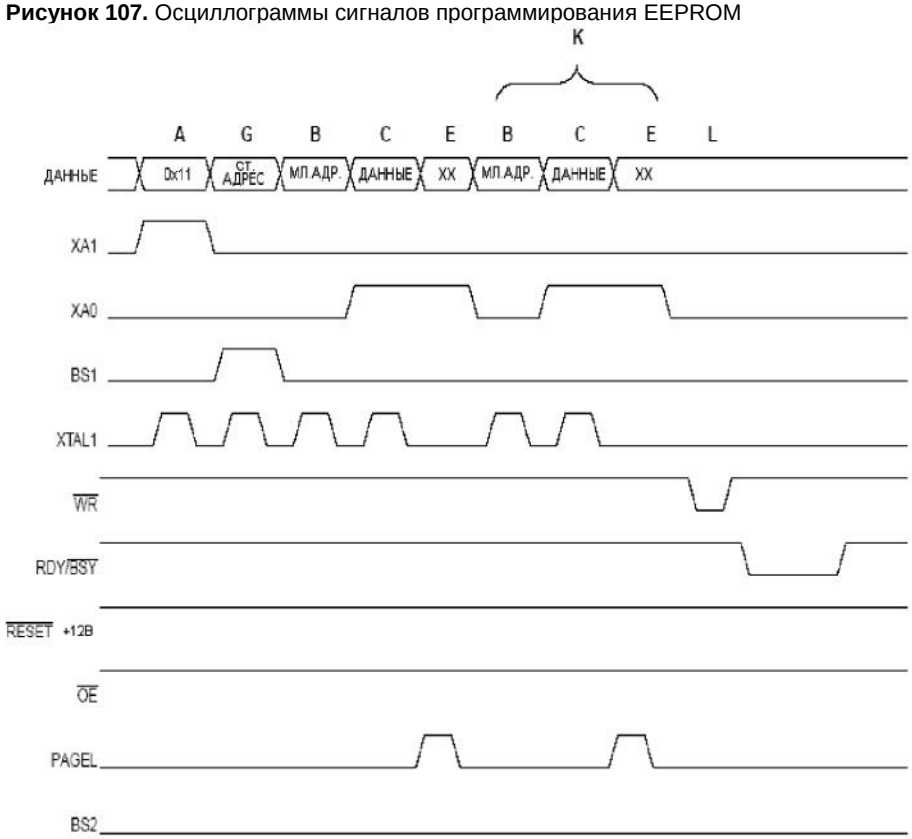
Прим.: "XX" означает, что не имеет значения, какие данные будут присутствовать. Указанные на рисунке символы соответствуют рассмотренному выше алгоритму.

Программирование EEPROM

EEPROM имеет страничную организацию (см. табл. 90). Во время программирования EEPROM программируемые данные размещаются в страничном буфере. Такая организация позволяет записать сразу одну страницу. Алгоритм программирования памяти данных EEPROM следующий (см. "Программирование флэш-памяти " для изучения подробностей загрузки команды, адреса и данных):

I. A: Загрузка команды "0001 0001".
II. G: Загрузка старшего байта адреса ($00 - $FF).
III. B: Загрузка младшего байта адреса ($00 - $FF).
IV. C: Загрузка данных ($00 - $FF).
V. E: Запись данных (положительный фронт на PAGEL).
VI. K: Повторяем 3-5 до заполнения всего буфера.
VII. L: Программирование страницы EEPROM :
1. Установка BS1 = "0".
2. Формируем отрицательный импульс на WR. Этим инициируется программирование страницы EEPROM . RDY/BSY переходит в низкое состояние.
3. Ожидаем появление лог. 1 на RDY/BSY перед программированием следующей страницы (см. осциллограммы на рисунок 107).

Рисунок 107. Осциллограммы сигналов программирования EEPROM



Чтение флэш-памяти

Алгоритм чтения флэш-памяти следующий (подробности по загрузке команд и адреса "Программирование флэш-памяти" страница 229):

1. A: Загрузка команды "0000 0010".
2. G: Загрузка старшего байта адреса ($00 - $FF).
3. B: Загрузка младшего байта адреса ($00 - $FF).
4. Установка OE = "0" и BS1 = "0". C линий данных может быть считан младший байт.
5. Установка BS1 = "1". C линий данных может быть считан старший байт.
6. Установка OE = "1".

Чтение EEPROM

Алгоритм чтения EEPROM следующий (см. "Программирование флэш-памяти " для изучения подробностей загрузки команды и адреса страница 229):
1. A: Загрузка команды "0000 0011".
2. G: Загрузка старшего байта адреса ($00 - $FF).
3. B: Загрузка младшего байта адреса ($00 - $FF).
4. Установка OE = "0" и BS1 = "0". Байт данных EEPROM может быть считан с линий данных.
5. Установка OE = "1".

Программирование младших конфигурационных бит

Алгоритм программирования младших конфигурационных бит следующий (подробности по загрузке команд и адреса см. в "Программирование флэш-памяти"):
1. A: Загрузка команды "0100 0000".
2. C: Загрузка младшего байта данных. Значение бита n = "0"/"1" соответствует программированию/стиранию конфигурационного бита.
3. Установка BS1 = "0" и BS2 = "0".
4. Формируем отрицательный фронт на WR и ожидаем появление лог.1 на RDY/BSY.

Программирование старших конфигурационных бит

Алгоритм программирования старших конфигурационных бит следующий (подробности по загрузке команд и адреса см. в "Программирование флэш-памяти"):
1. A: Загрузка команды "0100 0000".
2. C: Загрузка младшего байта данных. Значение бита n = "0"/"1" соответствует программированию/стиранию конфигурационного бита.
3. Установка BS1 = "1" и BS2 = "0". Этим выбирается старший байт данных.
4. Формируем отрицательный фронт на WR и ожидаем появление лог. 1 на RDY/BSY.
5. Установка BS1 = "0". Этим выбирается мл. байт данных.

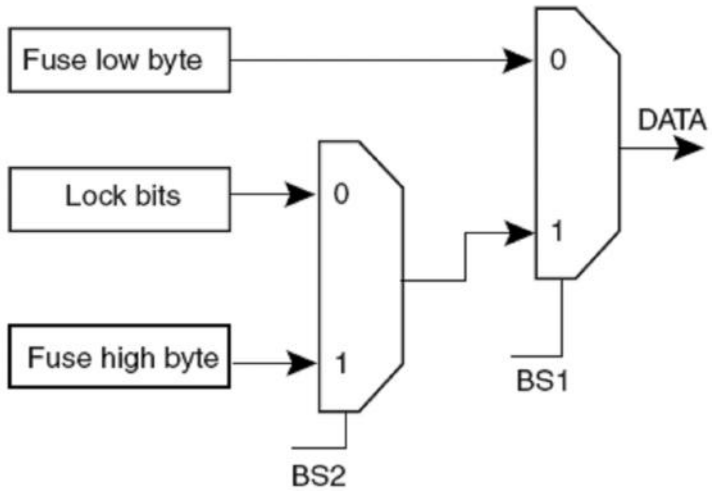
Программирование бит защиты

Алгоритм программирования бит защиты следующий (подробности по загрузке команд и адреса см. в "Программирование флэш-памяти"):
1. A: Загрузка команды "0010 0000".
2. C: Загрузка младшего байта данных. Бит n = "0" программирует бит защиты.
3. Формируем отрицательный фронт на WR и ожидаем появление лог. 1 на RDY/BSY.
Биты защиты стираются выполнением только командой стирания кристалла.

Чтение конфигурационных бит и бит защиты

Алгоритм чтения конфигурационных бит и бит защиты следующий (подробности по загрузке команд и адреса см. в "Программирование флэш-памяти"):
1. A: Загрузка команды "0000 0100".
2. Установка OE = "0", BS2 = "0" и BS1 = "0". Состояние младших конфигурационных бит может быть считано с линий данных ("0" означает запрограммированное состояние).
3. Установка OE = "0", BS2 = "1" и BS1 = "1". Состояние старших конфигурационных бит может быть считано с линий данных ("0" означает запрограммированное состояние).
4. Установка OE = "0", BS2 = "0" и BS1 = "1". Состояние бит защиты может быть считано с линий данных ("0" означает запрограммированное состояние).
5. Установка OE = "1".

Рисунок 108. Схема считывания конфигурационных бит и бит защиты под управлением сигналов BS1, BS2



Чтение сигнатурных байт

Алгоритм чтения сигнатурных байт следующий (подробности по загрузке команд и адреса см. в "Программирование флэш-памяти"):
1. A: Загрузка команды "0000 1000".
2. B: Загрузка младшего адресного байта ($00 - $02).
3. Установка OE = "0", BS1 = "0". Выбранный сигнатурный байт может быть считан с линий данных.
4. Установка OE = "1".

Чтение калибровочного байта

Алгоритм чтения калибровочного байта следующий (подробности по загрузке команд и адреса см. в "Программирование флэш-памяти")
1. A: Загрузка команды "0000 1000".
2. B: Загрузка младшего байта данных.
3. Установка OE = "0", BS1 = "1". Калибровочный байт может быть считан с линий данных.
4. Установка OE = "1".

Характеристики параллельного программирования

Рисунок 109. Временная диаграмма параллельного программирования: общие требования к временной диаграмме

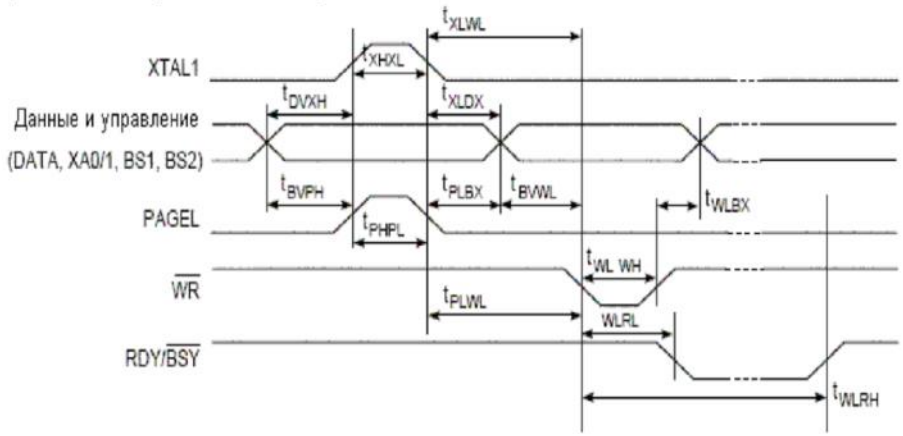
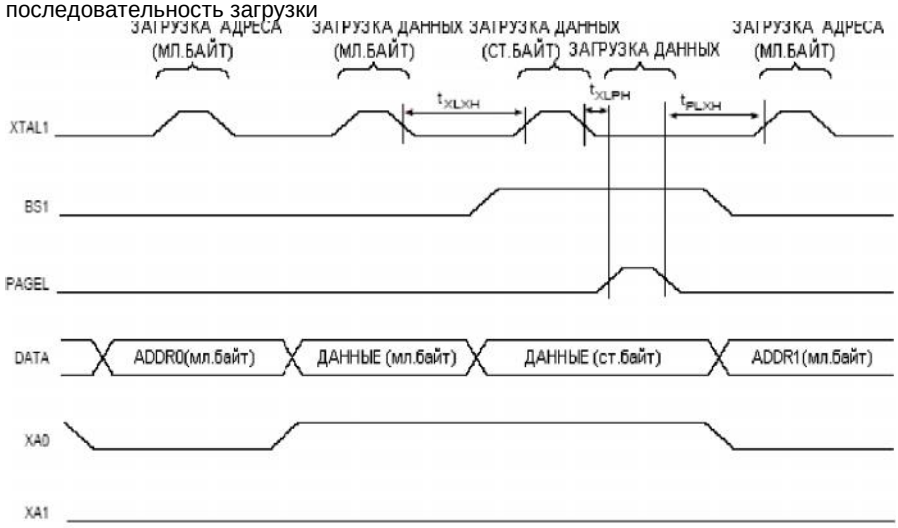
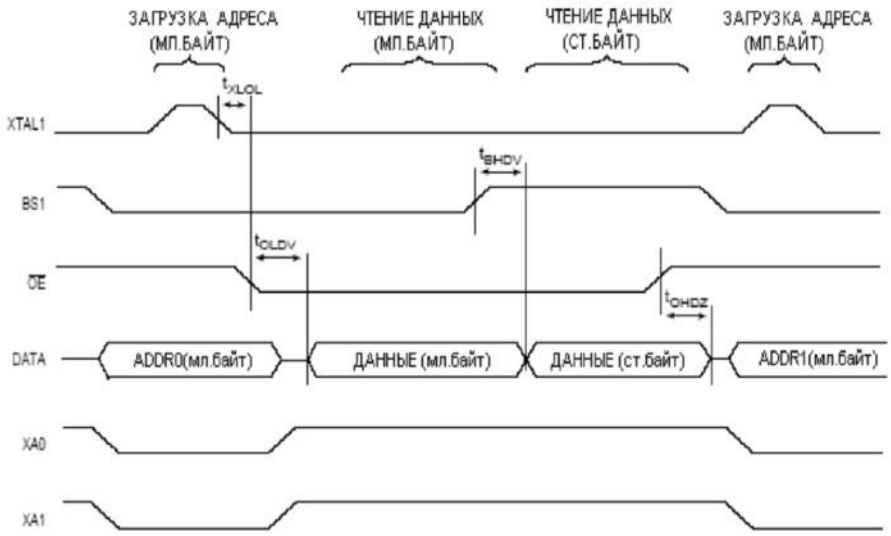


Рисунок 110. Временная диаграмма параллельного программирования: последовательность загрузки



Прим.: требования к временной диаграмме, показанные на рисунке 109 (в т.ч. tDVXH, tXHXL и tXLDX), также применимы и к операции загрузки

Рисунок 111. Временная диаграмма параллельного программирования: последовательность чтения (в пределах одной страницы)



Прим.: требования к временной диаграмме, показанные на рисунке 109 (в т.ч. tDVXH, tXHXL и tXLDX), также применимы и к операции загрузки.

Таблица 95. Характеристики параллельного программирования при VCC = 5В ± 10%

Обозначение	Параметр	мин.	ном.	макс.	Ед.изм.
V_{PP}	Напряжение программирования	11,5		12,5	В
I_{PP}	Ток программирования			250	мкА
t_{DVXH}	Задержка до появления лог. 1 на XTAL1 для действительности данных и управления	67			нс
t_{XLXH}	Длительность положительного фронта на XTAL1	200			нс
t_{XHXL}	Длительность единичного импульса на XTAL1	150			нс
t_{XLDX}	Удержание данных и управления после установки лог. 0 на XTAL1	67			нс
t_{XLWL}	Время между появлением лог. 0 на XTAL1и WR	0			нс
t^{XLPH}	Время между появлением лог. 0 на XTAL1 и лог. 1 на PAGEL	0			нс
t_{PLXH}	Время между появлением лог. 0 на PAGEL и лог. 1 на XTAL1	150			нс
t_{BVPH}	Действительность BS1 до появления лог. 1 на PAGEL	67			нс
t_{PHPL}	Длительность единичного импульса на PAGEL	150			нс
t_{PLBX}	Удержание BS1 после появления лог. 0 на PAGEL	67			нс
t_{WLBX}	Удержание BS2/1 после подачи лог. 0 на WR	67			нс
t_{PLWL}	Время между появлением лог. 0 на PAGEL и лог. 0 на WR	67			нс
t_{BVWL}	Действительность BS1 до появления лог. 0 на WR	67			нс
t_{WLWH}	Длительность низкого уровня импульса WR	150			нс
t_{WLRL}	Время между появлением лог. 0 на WR и RDY/BSY	0		1	мкс
t_{WLRH}	Время между появление лог. 0 на WR и лог. 1 на RDY/BSY(1)	3,7		4,5	мс
t_{WLRH_CE}	Время между появлением лог.0 на WR и лог. 1 на RDY/BSY для стирания кристалла (Chip Erase)(2)	7,5		9,0	мс
t_{XLOL}	Время между появлением лог.0 на и лог. 0 на OE	0			нс

t_{BVDV}	Время между действительностью BS1 и данными	0		250	нс
t_{OLDV}	Задержка на появление действительных данных после установки лог. 0 на OE			250	нс
t_{OHDZ}	Задержка на переход в высокоимпедансное состояние линий			250	нс
	данных после установки лог. 1 на OE				

Прим.:
1. tWLRH относится к командам записи флэш-памяти, ЭСППЗУ, конфигурационных бит и бит защиты.
2. tWLRH_CE относится к команде стирания кристалла (Chip Erase).

Последовательное программирование

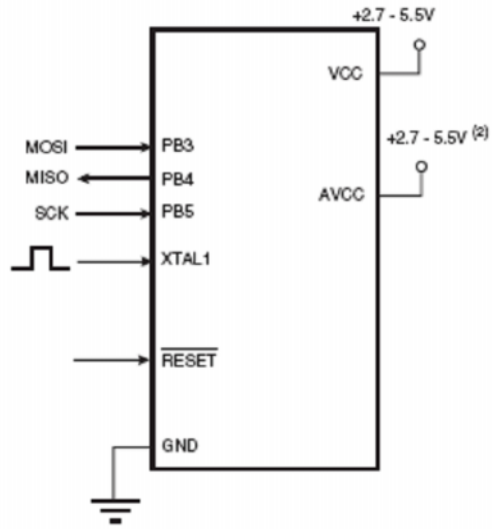
Флэш-память и EEPROM могут быть запрограммированы через последовательный интерфейс SPI, когда вход RESET переведен в низкое состояние. Последовательный интерфейс состоит из следующих сигналов: SCK, MOSI (вход) и MISO (выход). После подачи низкого уровня на вход RESET необходимо выполнить инструкцию разрешения программирования. В таблице 96 представлено описание сигналов программирования. Обратите внимание, что не все выводы последовательного программирования совпадают с выводами внутреннего интерфейса SPI. Также следует отметить, что повсюду при описании последовательного программирования используются наименования MOSI и MISO для описания последовательного ввода и вывода данных, соответственно.

Расположение выводов последовательного программирования через SPI

Таблица 96. Выводы интерфейса SPI при последовательном программировании

Обозначение	Вывод	Направление	Описание
MOSI	PB3	ввод	Последовательный ввод данных
MISO	PB4	вывод	Последовательный вывод данных
SCK	PB5	ввод	Синхронизация последовательной связи

Рисунок 112. Последовательное программирование и проверка(1)



Прим.:
1. Если микроконтроллер тактируется внутренним генератором, то нет необходимости подключать тактовый источник к выводу XTAL1.
2. VCC - 0.3В < AVCC < VCC + 0.3В, но AVCC должен находится в пределах 2.7 - 5.5В.

Во время программирования EEPROM функция стирания выполняется автоматически (только в режиме последовательного программирования) и, поэтому, нет необходимости первоначально выполнять команду "Стирание кристалла". Выполнение команды "Стирание кристалла" приводит к заполнению памяти программ и EEPROM кодом $FF.

Параметры тактового сигнала зависят от настроек синхронизации микроконтроллера конфигурационными битами CKSEL. Длительности высокого и низкого уровней тактового сигнала (SCK) должны отвечать следующим условиям:

Длительность низкого уровня:
- больше двух тактов ЦПУ, если fck < 12 МГц;
- больше трех тактов ЦПУ, если fck i 12 МГц.

Длительность высокого уровня:
- больше двух тактов ЦПУ, если fck < 12 МГц;
- больше трех тактов ЦПУ, если fck i 12 МГц.

Алгоритм последовательного программирования через SPI

Во время последовательной записи в ATmega8 данные тактируется нарастающим фронтом SCK. Во время чтения данных из ATmega8 данные тактируются падающим фронтом SCK.
Временная диаграмма представлена на рисунке 113.

Для программирования и проверки памяти ATmega8 в режиме последовательного программирования через SPI рекомендуется придерживаться следующей последовательности (см. четырехбайтный формат в таблице 98):
1. Последовательность подачи питания: подать напряжение питания между VCC и GND, когда на входах RESET и SCK присутствует лог. 0. В некоторых системах, программатор не может гарантировать, что SCK = 0 при подаче питания. В этом случае необходимо сформировать положительный импульс на RESET длительностью не менее двух тактов ЦПУ после того, как SCK принял низкое состояние.
2. Пауза не менее 20 мс и разрешение последовательного программирования путем записи команды разрешения последовательного программирования через вход MOSI.
3. Инструкции последовательного программирования не выполняются, если последовательная связь не вошла в синхронизацию. Вход в синхронизацию индицирует прием значения второго байта ($53) при записи третьего байта инструкции разрешения последовательного программирования. В зависимости от того корректно или нет принятое значение передаются все четыре байта инструкции. Если принятое значение не равно $53, то формируется положительный импульс на входе RESET и вводится новая команда разрешения последовательного программирования.
4. Флэш-память программируется постранично. Размер страницы показан в таблице 89. Страница памяти загружается побайтно, при этом в инструкции "загрузки страницы памяти программ" указываются данные и адрес в семи младших разрядах. Чтобы гарантировать корректность загрузки страницы сначала необходимо записать младший байт, а затем старший байт данных по каждому адресу. Запись страницы памяти программ инициируется вводом инструкции "запись страницы памяти программ", где в 9-ти старших разрядах указывается адрес страницы. Если опрос не используется, то программист должен предусмотреть задержку не менее tWD_FLASH перед вводом новой страницы (см. табл. 97).
Прим.: если какая-либо другая команда, кроме опроса (чтения), вводится перед завершением любой операции записи (флэш-память, EEPROM, биты защиты, конфигурационные биты) программирование может завершиться некорректно.
5. Массив памяти EEPROM программируется побайтно, при этом, в инструкции записи указывается адрес и данные. Перед записью данных первоначально автоматически стирается адресуемая ячейка ЭСППЗУ. Если опрос не используется, то программист должен предусмотреть задержку tWD_EEPROM перед вводом следующего байта (см. табл. 97). По стирания памяти микроконтроллера необходимо записывать только данные неравные $FF.
6. Любую ячейку памяти можно проверить использованием инструкции чтения, которая возвращает содержимое ячейки по указанному адресу путем последовательной передачи на выходе MISO.
7. По завершении программирования вход RESET должен быть переведен в высокое состояние для возобновления нормальной работы.
8. Последовательность снятия питания (при необходимости): установка RESET = "1", отключить питание VCC.

Опрос данных флэш-памяти

После того, как страница полностью запрограммирована во флэш-память, при чтении по адресам в пределах запрограммированной страницы возвращается $FF. Микроконтроллер готов к записи новой страницы, если запрограммированное значение считано корректно. Это используется для определения момента, когда может быть загружена следующая страница.
Обратите внимание, что запись выполняется всей страницы одновременно и любой адрес в пределах страницы может использоваться для опроса. Опрос данных флэш-памяти не действует для значения $FF, т.к. при записи этого значения пользователь может не вводить задержку tWD_FLASH перед программированием новой страницы.

Данная возможность объясняется тем, что очищенная память микроконтроллера содержит $FF во всех ячейках. Значение tWD_FLASH представлено в таблице 97.

Опрос данных EEPROM

При чтении значения по адресу, который использовался для записи нового байта и последующего его программирования в EEPROM , возвращается значение $FF. В это же время, микроконтроллер готов к записи нового байта, если запрограммированное значение корректно считывается. Это используется для определения момента, когда может быть осуществлена запись следующего байт. Данное не распространяется на значение $FF, но программист должен обратить внимание на следующее: поскольку очищенная память заполнена $FF по всем адресам, то программирование ячейки значением $FF может быть пропущено. Пропуск нельзя делать, если EEPROM перепрограммируется без предварительного стирания всей памяти. В этом случае, значение $FF нельзя использовать для опроса данных и программист должен предусмотреть задержку не менее tWD_EEPROM перед программированием следующего байта. В таблице 97 представлен о значение tWD_EEPROM.

Таблица 97. Минимальные длительности задержек перед записью очередной ячейки флэш-памяти и EEPROM

Обозначение	Минимальная задержка
t_{WD_FUSE}	4.5 ms
t_{WD_FLASH}	4.5 ms
t_{WD_EEPROM}	9.0 ms
t_{WD_ERASE}	9.0 ms

Рисунок 113. Осциллограммы сигналов последовательного программирования интерфейса SPI

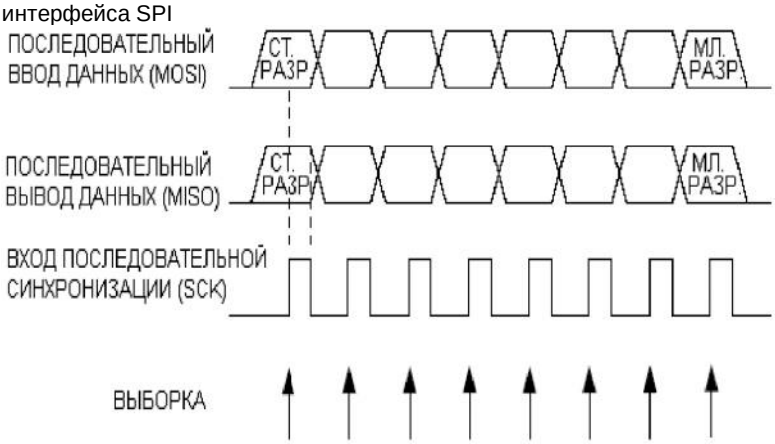


Таблица 98. Набор инструкций последовательного программирования через SPI

Инструкция	Формат инструкции				Функция
	Байт 1	Байт 2	Байт 3	Байт 4	
Разрешение программирования	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Разрешение последовательного программирования после подачи лог. 0 на RESET.
Стирание кристалла	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Стирание EEPROM и флэш-памяти
Чтение памяти программ	0010 **H**000	0000 **aaaa**	**bbbb bbbb**	**oooo oooo**	Чтение старшего (H=1) или младшего (H=0) байта данных o из памяти программ по адресу a:b.
Загрузка страницы памяти программ	0100 **H**000	0000 xxxx	xxx**b bbbb**	**iiii iiii**	Запись старшего (H=1) или младшего (H=0) байта данных i в страницу памяти программ по адресу b. Мл. байт данных должен быть загружен перед старшим байтом по тому же адресу.
Запись страницы памяти программ	0100 1100	0000 **aaaa**	**bbbb** xxxx	xxxx xxxx	Запись страницы памяти программ по адресу a:b.
Чтение EEPROM	1010 0000	00xx xxx**a**	**bbbb bbbb**	**oooo oooo**	Чтение данных o из EEPROM по адресу a:b.
Запись EEPROM	1100 0000	00xx xxx**a**	**bbbb bbbb**	**iiii iiii**	Запись данных i в EEPROM по адресу a:b.
Чтение бит защиты	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	Чтение бит защиты. "0" - запрограммирован, "1" - не запрограммирован. См. табл. 85.
Запись бит защиты	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	Запись бит защиты. Запись "0" приводит к программированию бита защиты. См. табл. 85
Чтение сигнатурного байта	0011 0000	00xx xxxx	xxxx xx**bb**	**oooo oooo**	Чтение сигнатурного байта o по адресу b.
Запись конфигурационных бит	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Установите "0" для программирования, "1" для стирания. См. табл. 88.
Запись старших конфигурационных бит	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Установите "0" для программирования, "1" для стирания. См. табл. 87.
Чтение конфигурационных бит	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Чтение конфигурационных бит. "0" - запрограммирован, "1" - не запрограммирован. См. табл. 88.
Чтение старших конфигурационных бит	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Чтение старших конфигурационных бит. "0" = запрограммирован, "1" = не запрограммирован. См. табл. 87.
Чтение калибровочного байта	0011 1000	00xx xxxx	0000 00**bb**	**oooo oooo**	Чтение калибровочного байта o по адресу b.

Прим.:
a - адрес старших разрядов;
b - адрес младших разрядов;
H - 0 - мл. байт, 1 - ст. байт;
o - вывод данных;
i - ввод данных;
x - произвольное значение.

Характеристики последовательного программирования через интерфейс SPI

Характеристики модуля SPI представлены в разделе "**Временные характеристики интерфейса SPI**" на странице 246.

Электрические характеристики

Предельно-допустимые параметры*

Рабочая температура	-55°C...+125°C
Температура хранения	-65°C...+150°C
Напряжение на любом выводе по отношению к общему питания, кроме RESET	-1.0В ... VCC+0.5В
Напряжение на выводе сброса RESET по отношению к общему	-1.0В ... +13.0В
Максимальное рабочее напряжение	6.0В
Постоянный ток через линию ввода-вывода	40.0 мА
Постоянный ток через выводы VCC и GND	200.0 мА

***ПРЕДУПРЕЖДЕНИЕ:**
выход за предельно допустимые параметры может вызвать перманентное повреждение микроконтроллера. Данные характеристики указывают на перегрузочные способности микроконтроллера, не следует их использовать как рабочие характеристики. Работа при условиях близким к предельно допустимым параметрам может повлиять на надежность работы микроконтроллера.

Статические характеристики

Если условия измерения не указаны, то полагается: Токр.ср. = -40°C…+85°C, VCC = 2.7В…5.5В.

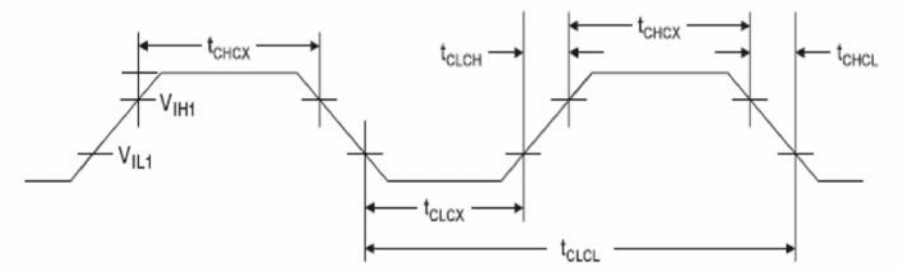
Обозн.	Параметр	Условия измерения	Мин.	Ном	Макс	Ед.из.
V_{IL}	Входное напряжение низкого уровня. Кроме выводов XTAL1 и RESET	V_{CC} = 2.7V - 5.5V	-0.5		0.2 VCC(1)	V
V_{IH}	Входное напряжение высокого уровня. Кроме выводов XTAL1 , RESET	V_{CC} = 2.7V - 5.5V	0.6 $V_{CC(2)}$		V_{CC} + 0.5	V
V_{IL1}	Входное напряжение низкого уровня, вывод XTAL1	V_{CC} = 2.7V - 5.5V	-0.5		0.1 $V_{CC(1)}$	V
V_{IH1}	Входное напряжение высокого уровня, вывод XTAL1	V_{CC} = 2.7V - 5.5V	0.8 $V_{CC(2)}$		V_{CC} + 0.5	V
V_{IL2}	Входное напряжение низкого уровня вывод сброса RESET	V_{CC} = 2.7V - 5.5V	-0.5		0.2 V_{CC}	V
V_{IH2}	Входное напряжение высокого уровня вывод сброса RESET	V_{CC} = 2.7V - 5.5V	0.9 $V_{CC(2)}$		V_{CC} + 0.5	V
V_{IL3}	Входное напряжение низкого уровня вывод сброса RESET линия ввода-вывода	V_{CC} = 2.7V - 5.5V	-0.5		0.2 V_{CC}	V
V_{IH3}	Входное напряжение высокого уровня вывод сброса RESET линия ввода-вывода	V_{CC} = 2.7V - 5.5V	0.6 $V_{CC(2)}$ 0.7 $V_{CC(2)}$		V_{CC} + 0.5	V
V_{OL}	Выходное напряжение низкого уровня (3)(порты B,C,D)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 3V			0.7 0.5	V V
V_{OH}	Выходное напряжение высокого уровня (4)(порты B,C,D)	I_{OH} = -20 mA, V_{CC} = 5V I_{OH} = -10 mA, V_{CC} = 3V	4.2 2.2			V V
I_{IL}	Входной ток утечки через линию ввода-вывода	Vcc = 5.5В, лог. 0 (абс. значение)			1	µA
I_{IH}	Входной ток утечки через линию ввода-вывода	Vcc = 5.5В, лог. 1 (абс. значение)			1	µA
R_{RST}	Сопротивление подтягивающего резистора на входе сброса		30		80	kΩ

R_{pu}	Сопротивление подтягивающего резистора на линиях ввода-вывода		20		50	kΩ
I_{CC}	Потребляемый ток	4 МГц, VCC = 3В, активный режим (ATmega8L)			5	mA
		8МГц, VCC = 5В, активный режим (ATmega8)			15	mA
		4 МГц, VCC = 3В, режим холостого хода (ATmega8L)			2	mA
		8 МГц, VCC = 5В, режим холостого хода (ATmega8)			7	mA
	Режим выключения (Powerdown)(5)	Стор. таймер включен, VCC= 3В			28	µA
		Стор. таймер отключен, VCC = 3В			3	µA
V_{ACIO}	Входное напряжение смещения аналогового компаратора	V_{CC} = 5V V_{in} = V_{CC}/2			40	mV
I_{ACLK}	Входной ток утечки аналогового компаратора	V_{CC} = 5V V_{in} = V_{CC}/2	-50		50	nA
t_{ACPD}	Задержка распространению сигнала в аналоговом компараторе	V_{CC} = 2.7V V_{CC} = 5.0V		750 500		ns

Примечания:
1. "Макс." означает наибольшее значение напряжения, приложенное к выводу, которое гарантированно распознается как логический 0.
2. "Мин." означает наименьшее значение напряжения, приложенное к выводу, которое гарантированно распознается как логическая 1.
3. Несмотря на то, что каждая линия ввода-вывода может быть нагружена втекающим током более, чем показано в условиях испытания (20 мА при питании VCC = 5В, 10 мА при питании VCC = 3В) в статическом состоянии (не переходном), необходимо учесть следующее:
Корпуса PDIP :
1] Суммарные токи IOL всех линий ввода-вывода не должны превышать 300 мА.
2] Суммарные токи IOL всех линий ввода-вывода C0 – C5 не должны превышать 100 мА.
3] Суммарные токи IOL всех линий ввода-вывода B0 - B7,C6, D0 - D7, XTAL2 не должны превышать 200 мА.
Корпуса TQFP и QFN/MLF:
1] Суммарные токи IOL всех линий ввода-вывода не должны превышать 300 мА.
2] Суммарные токи IOL всех линий ввода-вывода C0 – C5 не должны превышать 100 мА
Если IOL превышает условия испытания, то VOL может также увеличиться. Для выводов не гарантируется втекающий ток выше приведенного в условиях испытания.
4. Несмотря на то, что каждая линия ввода-вывода может быть нагружена вытекающим током больше, чем показано в условиях испытания (20 мА при Vcc = 5В, 10 мА при Vcc = 3В) в статическом состоянии (не переходном), необходимо учесть следующее:
Корпуса PDIP:
1] Суммарные токи IOH всех линий ввода-вывода не должны превышать 300 мА.
2] Суммарные токи IOH всех линий ввода-вывода C0 – C5 не должны превышать 100 мА.
3] Суммарные токи IOH всех линий ввода-вывода B0 - B7,C6, D0 - D7, XTAL2 не должны превышать 200 мА.
Корпуса TQFP и QFN/MLF:
1] Суммарные токи IOH всех линий ввода-вывода не должны превышать 300 мА.
2] Суммарные токи IOH всех линий ввода-вывода C0 – C5 не должны превышать 100 мА
Если IOH превышает условия испытания, то VOH также может выйти за указанные значения. Для выводов не гарантируется вытекающий ток выше приведенного в условиях испытания.
5. Минимальное VCC для режима выключения: 2.5В.

Требования к характеристикам внешнего тактового сигнала

Рисунок 114. Осциллограмма внешнего тактового сигнала



Параметры внешнего тактового сигнала

Таблица 99. Параметры внешнего тактового сигнала

Обозначение	Параметр	V_{CC} = 2.7В...5.5В		V_{CC} = 4.5В...5.5В		Ед.изм.
		мин.	макс.	мин.	макс.	
$1/t_{CLCL}$	Частота генератора	0	8	0	16	МГц
t_{CLCL}	Период синхронизации	125		62,5		нс
t_{CHCX}	Длительность единичного импульса	50		25		нс
t_{CLCX}	Длительность нулевого импульса	50		25		нс
t_{CLCH}	Длительность нарастающего фронта		1,6		0,5	мкс
t_{CHCL}	Длительность падающего фронта		1,6		0,5	мкс
Δt_{CLCL}	Разброс периодов смежных импульсов		2		2	%

Таблица 100. Типичные частоты при тактировании от внешней RC-цепи

R [kΩ](1)	C [pF]	f(2)
33	22	650 kHz
10	22	2.0 MHz

Прим.:
1. Сопротивление R должно находиться в пределах 3 кОм…100 кОм, а емкость не менее 20 пФ. Значения C представлены в таблице с учетом емкости вывода микроконтроллера. Емкость вывода может варьироваться в зависимости от типа корпуса.
2. Частота будет меняться в зависимости от типа корпуса и расположения платы.

Характеристики двухпроводного последовательного интерфейса

В таблице 101 описываются требования к устройствам, подключаемых к двухпроводной последовательной шине. Двухпроводной последовательный интерфейс ATmega8 отвечает данным требованиям или превосходит их в указанных условиях.

Обозначения параметров показаны на рисунке 115.

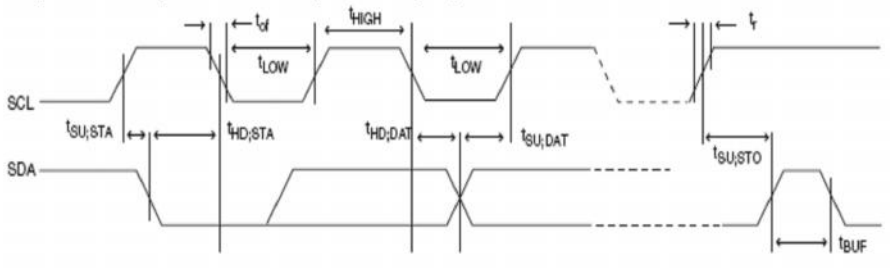
Таблица 101. Требования к двухпроводной последовательной шине

Обозн.	Параметр	Условия измерения	Мин.	Макс.	Ед.изм.
V_{IL}	Входное напряжение низкого уровня		-0.5	0.3 V_{CC}	В
V_{IH}	Входное напряжение высокого уровня		0.7 V_{CC}	V_{CC} + 0.5	В
$V_{гис}$ (1)	Гистерезис триггеров Шмитта		0.05 V_{CC} (2)	-	В
V_{OL} (1)	Выходное напряжение низкого уровня	Вытекающий ток 3 мА	0	0,4	В
t_r (1)	Время нарастания SDA и SCL		20 + 0.1Cb (3)(2)	300	нс
t_{of} (1)	Длительность спада с V_{IHmin} до V_{ILmax}	10 пФ < Cb < 400 пФ(3)	20 + 0.1Cb (3)(2)	250	нс
t_{SP} (1)	Длительность подавляемых импульсов входным фильтром		0	50(2)	нс
I_i	Входной ток каждой линии ввода-вывода	0.1 Vcc < Vi < 0.9 Vcc	-10	10	мкА
Ci (1)	Емкость каждой линии ввода-вывода		-	10	пФ
f_{SCL}	Частота синхронизации SCL	f_{CK} (4)>макс (16f_{SCL}, 250КГц)(5)	0	400	кГц
Rp	Значение подтягивающего резистора	f_{SCL} ≤100 кГц	V_{CC}-0,4В / 3 мА	1000 нс / - Cb	Ом
		f_{SCL} > 100 кГц	V_{CC}-0,4В / 3 мА	300 нс / - Cb	Ом
$t_{HD;STA}$	Время удержания условия СТАРТ (повторный старт)	f_{SCL} ≤100 кГц	4.0	-	мкс
		fSCL > 100 кГц	0.6	-	мкс
t_{LOW}	Длительность нулевого импульса SCL	f_{SCL} ≤100 кГц(6)	4.7	-	мкс
		f_{SCL} > 100 кГц(7)	1.3	-	мкс
t_{HIGH}	Длительность единичного импульса SCL	f_{SCL} ≤100 кГц	4.0	-	мкс
		f_{SCL} > 100 кГц	0.6	-	мкс
$t_{SU;STA}$	Время установки условия повторный старт	f_{SCL} ≤100 кГц	4.7	-	мкс
		f_{SCL} > 100 кГц	0.6	-	мкс
$t_{HD;DAT}$	Время удержания данных	f_{SCL} ≤100 кГц	0	3.45	мкс
		f_{SCL} > 100 кГц	0	0.9	мкс
$t_{SU;DAT}$	Время установки данных	f_{SCL} ≤100 кГц	250	-	нс
		fSCL > 100 кГц	100	-	нс
$t_{SU;STO}$	Время установки условия СТОП	f_{SCL} ≤100 кГц	4.0	-	мкс
		f_{SCL} > 100 кГц	0.6	-	мкс
t_{BUF}	Время освобождения шины между условиями СТОП и СТАРТ	f_{SCL} ≤100 кГц	4.7	-	мкс

Прим.:
1. Значение данного параметра у ATmega8 проверено не полностью.
2. Только для fSCL > 100 кГц
3. Cb - емкость одной линии шины в пФ.
4. fCK - тактовая частота ЦПУ

5. Данное требования относится ко всей работе двухпроводного последовательного интерфейса ATmega8. Другие устройства подключенные к двухпроводной последовательной шине могут отвечать только общим требованиям к fSCL.
6. Фактическая длительность низкого уровня, генерируемого двухпроводным последовательным интерфейсом ATmega128, составляет (1/fSCL - 2/fCK), таким образом, частота fCK должна быть больше 6 МГц для более точного выполнения требования к длительности низкого уровня при fSCL = 100 кГц.
7. Фактическая длительность низкого уровня, генерируемого двухпроводным последовательным интерфейсом ATmega128, составляет (1/fSCL - 2/fCK), таким образом, требования к длительности низкого уровня не строго выполняются для fSCL > 308 кГц, когда fCK = 8 МГц. Однако, микроконтроллеры ATmega128, подключенные к шине, могут связываться на полной скорости (400 кГц) между собой, а также с другими устройствами с надлежащим значением tLOW.

Рисунок 115. Временная диаграмма двухпроводной последовательной шины



Характеристики временной диаграммы SPI

Таблица 102. Параметры временной диаграммы SPI

	Описание	Режим	Мин.	Тип.	Макс	Ед.изм
1	Период SCK	ведущий		см. табл. 50		ns
2	Длительность низкого/высокого уровня SCK	ведущий		меандр		
3	Время нарастания/спада	ведущий		3.6		
4	Установка	ведущий		10		
5	Удержание	ведущий		10		
6	Вывод к SCK	ведущий		0.5 • tSCK		
7	SCK к выводу	ведущий		10		
8	SCK к выводу лог. 1	ведущий		10		
9	Низкий уровень SS к выводу	подчиненный		15		
10	Период SCK	подчиненный	4 • tck			
11	Длительность низкого/высокого уровня SCK(1)	подчиненный	2 • tck			
12	Время нарастания/спада	подчиненный			1.6	
13	Установка	подчиненный	10			
14	Удержание	подчиненный	10			
15	SCK к выводу	подчиненный		15		
16	SCK к высокому уровню SS	подчиненный	20			
17	Лог. 1 на SS к переходу в Z-состояние	подчиненный		10		
18	Низкий уровень на SS к SCK	подчиненный	2 • tck			

Прим.:
1. В режиме программирования через SPI минимальные длительности низкого/высокого уровней SCK должны быть следующие:
- 2 tCLCL для fCK < 12 МГц
- 3 tCLCL для fCK > 12 МГц

Рисунок 116. Требования к временной диаграмме интерфейса SPI (режим ведущего)

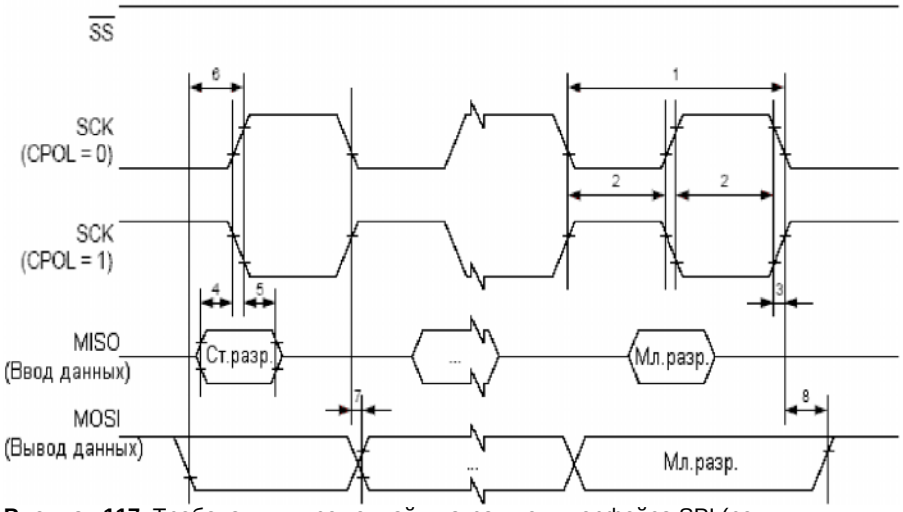
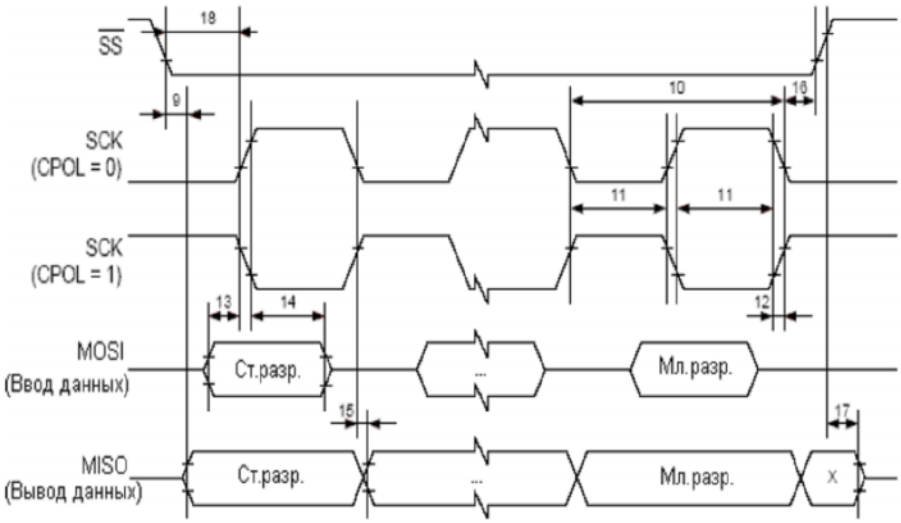


Рисунок 117. Требования к временной диаграмме интерфейса SPI (режим подчиненного)



Характеристики АЦП

Таблица 103. Характеристики АЦП

Обозн.	Параметр	Условия измерения	Мин.(1)	Тип.(1)	Макс.(1)	Ед.изм.
	Разрешение	Одиночное преобразование		10		бит
	Абсолютная погрешность	Одиночное преобразование; V=4В, V_{CC} = 4V Fсинхр.ацп=200кГц		1.75		мл.разр
		Одиночное преобразование; V_{REF} =4В ,V_{CC} = 4V Fсинхр.ацп=1МГц		3		мл.разр
	Интегральная нелинейность	Одиночное преобразование; V_{REF}=4В, V_{CC} = 4V Fсинхр.ацп=200кГц		0.75		мл.разр
	Дифференциальная нелинейность	Одиночное преобразование; V_{REF} =4В, V_{CC} = 4V Fсинхр.ацп=200кГц		0.5		мл.разр
	Ошибка усиления	Одиночное преобразование; V_{REF} =4В, V_{CC} = 4V Fсинхр.ацп=200кГц		1		мл.разр
	Ошибка смещения	Одиночное преобразование; V_{REF} =4В, V_{CC} = 4V Fсинхр.ацп=200кГц		1		мл.разр
	Время преобразования	Автоматическое преобразование	13		260	µs
	Тактовая частота		50		1000	kHz
AV_{CC}	Аналоговое напряжение питания		V_{CC} - 0.3(2)		V_{CC} + 0.3(3)	V
V_{REF}	Опорное напряжение		2.0		AV_{CC}	V
V_{IN}	Входное напряжение		GND		V_{REF}	V
	Вход. частотный диапазон			38		kHz
V_{INT}	Напряжение внутреннего опорного источника		2.3	2.56	2.7	V
R_{REF}	Сопротивление входа подключения опорного источника			32		kΩ
R_{AIN}	Сопротивление аналогового входа		55	100		MΩ

Прим.:
1. Приведенные значения носят предварительный характер. Фактические значения в состоянии определения.
2. Минимальное напряжение AVCC = 2.7 В.
3. Максимальное напряжение AVCC = 5.5 В
4. Максимальное время преобразования - 1/50kHz*25 = 0.5 мс.

Сводная таблица регистров

Адрес	Имя	Bit 7	Bit 6	Bit	Bit 4	Bit 3	Bit 2	Bit	Bit	стр.
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	11
0x3E (0x5E)	SP	–	–	–	–	–	SP1	SP	SP	13
0x3D (0x5D)	SP	SP	SP	SP	SP	SP	SP	SP	SP	13
0x3C (0x5C)	Резерв									
0x3B (0x5B)	GIC	INT	INT	–	–	–	–	IVSEL	IVC	49, 67
0x3A (0x5A)	GIF	INTF1	INTF0	–	–	–	–	–	–	68
0x39 (0x59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	72, 102, 122
0x38 (0x58)	TIF	OCF2	TOV2	ICF	OCF1A	OCF1B	TOV1	–	TOV	73, 103, 122
0x37 (0x57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	213
0x36 (0x56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWI	171
0x35 (0x55)	MCUCR	S	SM	SM	SM	ISC11	ISC10	ISC01	ISC00	33, 66
0x34 (0x54)	MCUCSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	41
0x33 (0x53)	TCCR0	–	–	–	–	–	CS02	CS0	CS0	72
0x32 (0x52)	TCNT0	Timer/Counter0								72
0x31 (0x51)	OSCCAL	Oscillator Calibration								31
0x30 (0x50)	SFIOR	–	–	–	–	ACME	PU	PSR2	PSR10	58, 75, 123, 193
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	97
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS1	CS1	100
0x2D (0x4D)	TCNT1H	Timer/Counter1 – Counter Register								101
0x2C (0x4C)	TCNT1L	Timer/Counter1 – Counter Register								101
0x2B (0x4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High byte								101
0x2A (0x4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low byte								101
0x29 (0x49)	OCR1BH	Timer/Counter1 – Output Compare Register B High byte								101
0x28 (0x48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low byte								101
0x27 (0x47)	ICR1H	Timer/Counter1 – Input Capture Register High byte								102
0x26 (0x46)	ICR1L	Timer/Counter1 – Input Capture Register Low byte								102
0x25 (0x45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS2	CS2	117
0x24 (0x44)	TCNT2	Timer/Counter2								119
0x23 (0x43)	OCR2	Timer/Counter2 Output Compare								119
0x22 (0x42)	ASSR	–	–	–	–	AS	TCN2UB	OCR2UB	TCR2UB	119
0x21 (0x41)	WDTCR	–	–	–	WDCE	WD	WDP2	WDP1	WDP0	43
0x20[1] (0x40)[1]	UBRRH	URSEL	–	–	–	UBRR[1				158
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	156
0x1F (0x3F)	EEARH	–	–	–	–	–	–	–	EEAR8	20
0x1E (0x3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	20
0x1D (0x3D)	EEDR	EEPROM Data								20
0x1C (0x3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	20
0x1B (0x3B)	Резерв									
0x1A (0x3A)	Резерв									
0x19 (0x39)	Резерв									
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	65
0x17 (0x37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB	65
0x16 (0x36)	PIN	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	65
0x15 (0x35)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	65
0x14 (0x34)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	65
0x13 (0x33)	PIN	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	65
0x12 (0x32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	65
0x11 (0x31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	65
0x10 (0x30)	PIN	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	65
0x0F (0x2F)	SPDR	SPI Data								131
0x0E (0x2E)	SPSR	SPI	WCOL	–	–	–	–	–	SPI2X	131
0x0D (0x2D)	SPCR	SPI	SP	DORD	MSTR	CPOL	CPHA	SPR1	SPR	129
0x0C (0x2C)	UD	USART I/O Data								153
0x0B (0x2B)	UCSRA	RX	TX	UDRE	F	DO	P	U2	MPCM	154
0x0A (0x2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB	155
0x09 (0x29)	UBRRL	USART Baud Rate Register								158
0x08 (0x28)	ACSR	AC	ACBG	AC	AC	ACI	ACI	ACIS1	ACIS0	194
0x07 (0x27)	ADMUX	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	205
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADFR	ADI	ADI	ADPS2	ADPS1	ADPS0	207
0x05 (0x25)	ADCH	ADC Data Register								208
0x04 (0x24)	ADCL	ADC Data Register								208
0x03 (0x23)	TWDR	Two-wire Serial Interface Data								173
0x02 (0x22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	174

Сводная таблица регистров(продолжение)

Адрес	Имя	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Стр
0x01 (0x21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	173
0x00 (0x20)	TWBR	Two-wire Serial Interface Bit Rate Register								171

Примечания:
1. Смотрите описание USART для деталей о том, как получить доступ к UBRRH и UCSRC.
2. Для совместимости с последующими версиями микроконтроллеров рекомендуется в резервные разряды записывать лог. 0. В резервные ячейки памяти ввода-вывода не рекомендуется выполнять запись.
3. Некоторые флаги состояния сбрасываются путем записи в них лог. 1. Обратите внимание, что инструкции CBI и SBI работают со всеми разрядами регистра ввода-вывода (чтение-модификация-запись). Инструкции CBI и SBI работают только с регистрами $00…$1F.