ATmega128, ATmega128L - 8-разрядный AVR-микроконтроллер с внутрисистемно программируемой флэш-памятью емкостью 128 кбайт

Ядро центрального процессорного устройства AVR

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

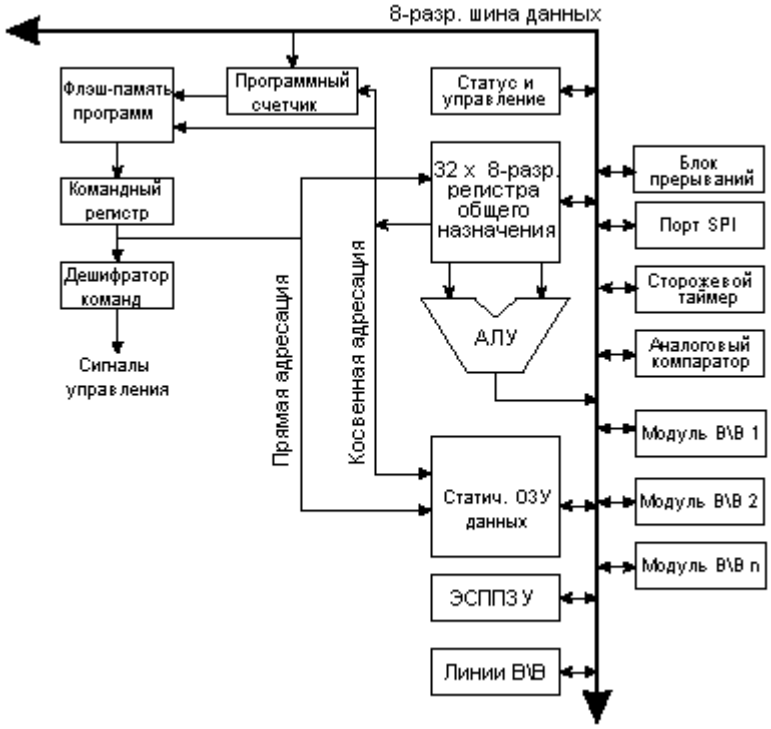


Рисунок 3 – Функциональная схема архитектуры AVR

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.

Для ветвления программы поддерживаются инструкции условных и безусловных переходов и вызовов процедур, позволяющих непосредственно адресоваться в пределах адресного пространства. Большинство инструкций представляют собой одно 16-разр. слово. Каждый адрес памяти программ содержит 16- или 32-разр. 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.

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 с непосредственной адресацией или может адресоваться как память данных, следующая за регистрами по адресам $20 - $5F. Кроме того, ATmega128 имеет пространство расширенного ввода-вывода по адресам $60 - $FF в статическом ОЗУ, для доступа к которому могут использоваться только процедуры ST/STS/STD и LD/LDS/LDD.

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 ATmega128 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:

Разряд	7	6	5	4	3	2	1	0	
	–	—	–	–	–	–	–	RAMPZ0	RAMPZ
Чтение/Запись	Чт	Чт	Чт	Чт	Чт	Чт	Чт	Чт/Зп	
Начальное значение	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 4 показывает структуру 32 рабочих регистров общего назначения в ЦПУ.

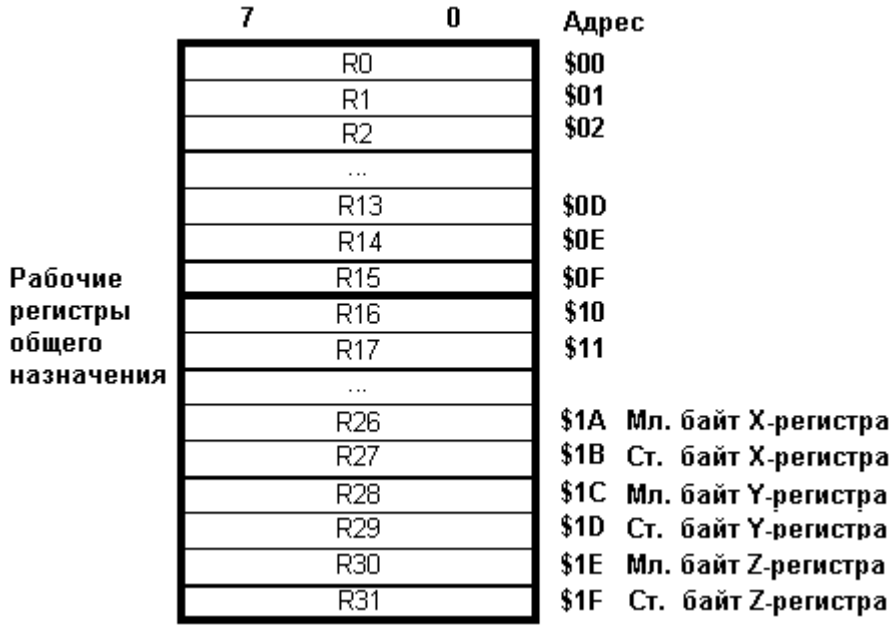


Рисунок 4 – Рабочие регистры общего назначения ЦПУ AVR

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

Как показано на рисунке 4, 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 5.

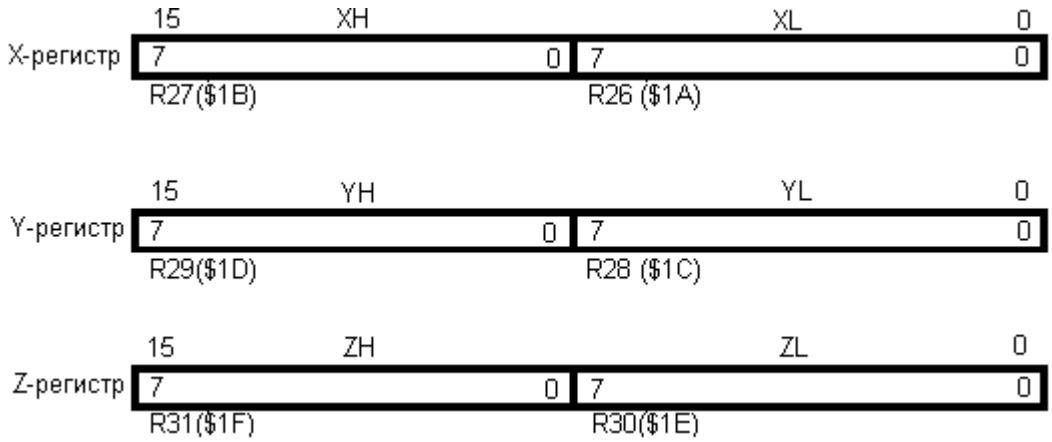


Рисунок 5 - X, Y и Z-регистры

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

Стек обычно используется для хранения временных данных, для хранения локальных переменных и для хранения адресов возврата при прерываниях и вызовах подпрограмм. Регистр указателя стека указывает на вершину стека. Обратите внимание на организацию стека, который направляется от старших в более младшие позиции статического ОЗУ. Это означает, что команда помещения в стек PUSH уменьшает значение указателя 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.

Регистр выбора Z-страницы ОЗУ – RAMPZ

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	

Разряд	7	6	5	4	3	2	1	0	
	I	**T**	**H**	**S**	**V**	**N**	**Z**	**C**	**SREG**
Чтение/запись	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	
Начальное знач.	0	0	0	0	0	0	0	0	

Разряды 7…2 – Зарезервированные разряды

Данные зарезервированные разряды считываются как 0. При записи в данные разряды необходимо записывать нули для совместимости с последующими микроконтроллерами.

Разряд 1 – RAMPZ0: Расширенный указатель страницы ОЗУ

Регистр RAMPZ обычно используется для указания той страницы ОЗУ размером 64 кбайт, к которой выполняется доступ через Z-указатель. Т.к. ATmega128 не поддерживает память на статическом ОЗУ размером свыше 64 кбайт, то данный регистр используется только для выбора страницы памяти программ, доступ к которой осуществляется с помощью инструкций ELPM/SPM. Различные установки бита RAMPZ0 имеют следующий результат:

RAMPZ0 = 0: Инструкции ELPM/SPM осуществляют доступ к памяти программ в диапазоне адресов $0000 - $7FFF (младшие 64 кбайт)

RAMPZ0 = 1: Инструкции ELPM/SPM выполняют доступ к памяти программ в диапазоне адресов $8000 - $FFFF (старшие 64 кбайт)

Обратите внимание, что действие инструкции LPM не зависит от установки RAMPZ.

Временная диаграмма выполнения инструкции

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

Рисунок 6 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.

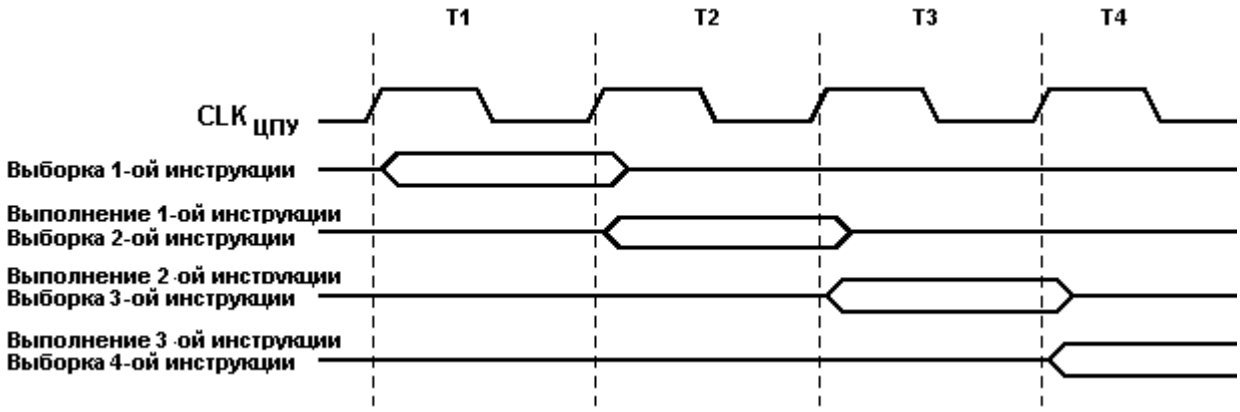


Рисунок 6 – Параллельные выборки и исполнения инструкций

Рисунок 7 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.

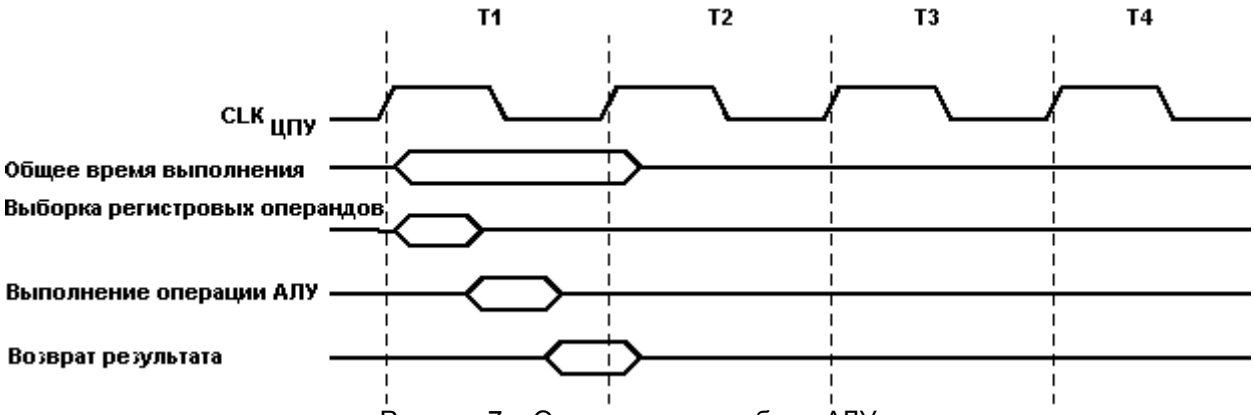


Рисунок 7 – Однотактность работы АЛУ

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 ЭСППЗУ.

```
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 ЭСППЗУ
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 ЭСППЗУ 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

В данном разделе описываются различные виды памяти ATmega128. В соответствии с гарвардской архитектурой память AVR-микроконтроллера разделена на две области: память данных и память программ. Кроме того, ATmega128 содержит память на ЭСППЗУ для

энергонезависимого хранения данных. Все три области памяти являются линейными и равномерными.

Внутрисистемно программируемая флэш-память программ ATmega128 содержит 128 кбайт внутренней внутрисистемно перепрограммируемой флэш-памяти для хранения программы. Поскольку все AVR-инструкции являются 16 или 32-разр., то флэш-память организована как 64 кбайт 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 ATmega128 является 16-разр., поэтому, позволяет адресоваться к 64 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”. В разделе “Программирование памяти” 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”.

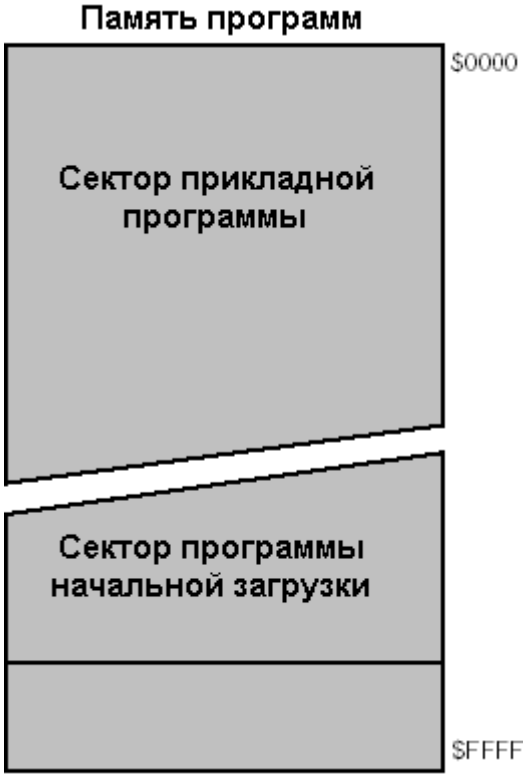


Рисунок 8- Карта памяти программ

Статическое ОЗУ памяти данных

ATmega128 поддерживает две различные конфигурации статического ОЗУ памяти данных (см. табл. 1).

Таблица 1 – Конфигурации памяти

Конфигурация	Встроенное статическое ОЗУ памяти данных	Внешнее статическое ОЗУ памяти данных

Нормальный режим	4096	до 64 кбайт
Режим совместимости с ATmega103	4000	до 64 кбайт

Рисунок 9 иллюстрирует организацию памяти на статическом ОЗУ у ATmega128.

ATmega128 – сложный микроконтроллер с большим числом периферийных устройств, которые управляются через 64 ячейки памяти, зарезервированных в кодах операций инструкций IN и OUT. Для расширенной области ввода-вывода в статическом ОЗУ по адресам $60 - $FF необходимо использовать только инструкции ST/STS/STD и LD/LDS/LDD. Область расширенного ввода-вывода не существует при переводе ATmega128 в режим совместимости с ATmega103.

В нормальном режиме первые 4352 ячейки памяти данных относятся к файлу регистров, памяти ввода-вывода, расширенной памяти ввода-вывода и встроенному статическому ОЗУ данных. В первых 32 ячейках расположен файл регистров, следующие 64 ячейки занимает стандартная память ввода-вывода, а за ними следуют 160 ячеек расширенной памяти ввода-вывода. Замыкают внутреннюю память данных 4096 ячеек внутреннего статического ОЗУ данных.

В режиме совместимости с ATmega103 первые 4096 ячеек памяти данных относятся к файлу регистров, памяти ввода-вывода и внутреннему статическому ОЗУ данных. В первых 32 ячейках расположен файл регистров, затем в 64 ячейках расположена стандартная память ввода-вывода и следующие 4000 ячейки занимает внутреннее ОЗУ данных.

Совместно с ATmega128 по выбору может использоваться статическое ОЗУ. Это статическое ОЗУ будет занимать оставшуюся часть от адресного пространства размером 64 кбайт. Данная часть начинается с адреса следующего за внутренним статическим ОЗУ. Файл регистров, память ввода-вывода, память расширенного ввода-вывода и внутреннее статическое ОЗУ занимают младшие 4352 байта в нормальном режиме и младшие 4096 байта в режиме совместимости с ATmega103. (отсутствует память расширенного ввода-вывода). Таким образом, при использовании внешней памяти размером 64 кбайт (65536 байт) из них будет доступно 61184 байта в нормальном режиме и 61440 байта в режиме совместимости с ATmega103. См. раздел “Интерфейс внешней памяти” для детального изучения методов использования внешней памяти.

Доступ к внешнему статическому ОЗУ осуществляется автоматически с помощью тех же инструкций, что и для внутреннего ОЗУ, если указанное значение адреса находится за пределами внутренней памяти данных. При адресации внутренней памяти сигналы чтения и записи внешней памяти (выводы PG0 и PG1) неактивны в процессе всего цикла доступа. Работа внешнего статического ОЗУ разрешается путем установки бита SRE в регистре MCUCR.

Доступ к внешнему статическому ОЗУ требует еще одного машинного цикла на байт по сравнению с доступом к внутреннему статическому ОЗУ. Это означает, что на выполнение команд LD, ST, LDS, STS, LDD, STD, PUSH и POP потребуется один дополнительный цикл. Если стек будет размещен во внешнем статическом ОЗУ, то, соответственно, вызов и возврат из подпрограмм и процедур обработки прерываний будет длиться на три машинных цикла дольше за счет помещения в стек и извлечения из стека двухбайтного счетчика программы и не использования во время доступа к внешней памяти преимущества конвейерного доступа к внутренней памяти. Если интерфейс внешнего статического ОЗУ используется с состояниями ожидания (со сниженным быстродействием), то однобайтный внешний доступ потребует 2, 3 или 4 дополнительных машинных цикла для 1, 2 и 3 состояний ожиданий, соответственно. Таким образом, вызов и возврат из прерываний и подпрограмм потребует еще 5, 7 и 9 машинных циклов (в отличие от значений приведенных в описании набора инструкций) для 1, 2 и 3 состояний ожидания, соответственно.

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 4096 байт внутреннего статического ОЗУ данных в ATmega128 доступны с помощью всех этих режимов адресации. Файл регистров описывается в разделе “Файл регистров общего назначения”.

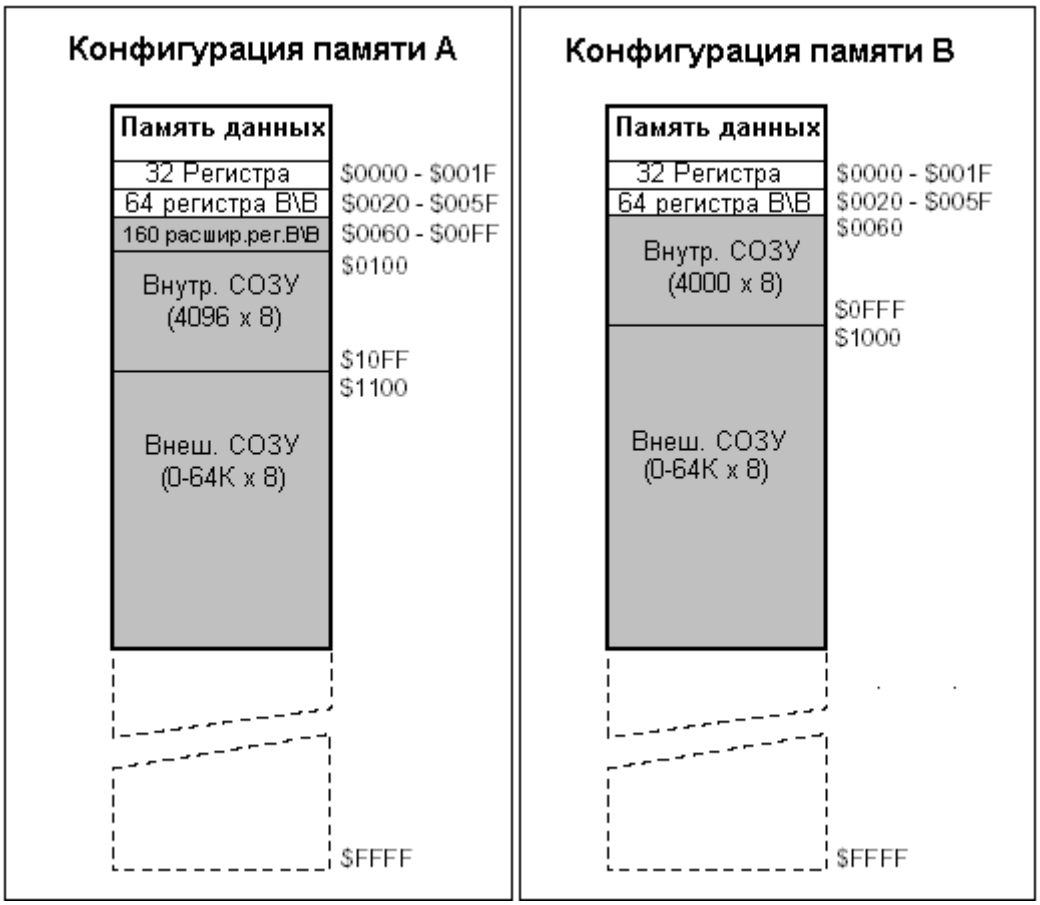


Рисунок 9 – Карта памяти данных

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 10.

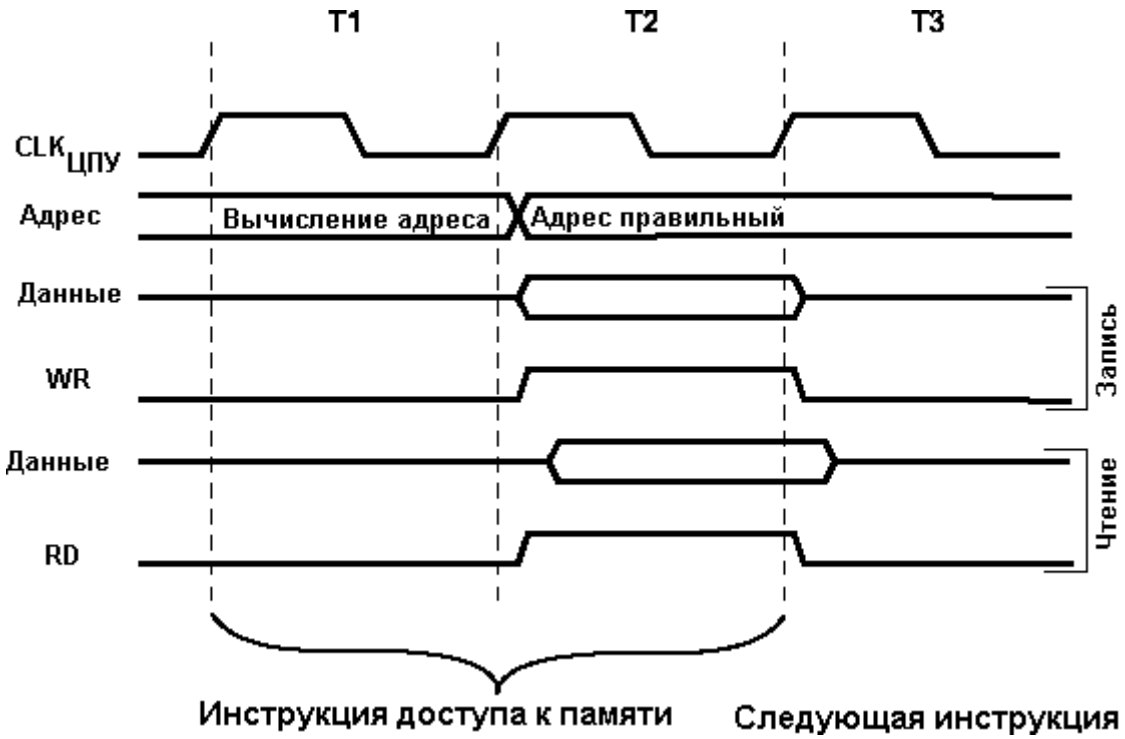


Рисунок 10 – Временная диаграмма доступа к встроенному статическому ОЗУ данных

Память данных на ЭСППЗУ

ATmega128 содержит 4 кбайт памяти данных на ЭСППЗУ. Она организована как отдельная область памяти данных, в которой один байт может быть записан и считан. ЭСППЗУ характеризуется износостойкостью 100000 циклов чтения/записи.

В разделе “Программирование памяти” содержится детальное описание программирование ЭСППЗУ через интерфейсы SPI, JTAG или параллельное программирование.

Чтение и запись ЭСППЗУ

Доступ к ЭСППЗУ осуществляется через специальные регистры, расположенные в пространстве ввода-вывода.

Время записи в ЭСППЗУ приведено в табл. 2. Функция самосинхронизации позволяет программно определить возможность записи следующего байта. Если код программы содержит инструкции записи в ЭСППЗУ, 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 ЭСППЗУ” для детального изучения методов разрешения данной проблемы.

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

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

Адресные регистры ЭСППЗУ – EEARH и EEARL

Разряд	15	14	13	12	11	10	9	8	
	–	–	–	–	EEAR11	EEAR10	EEAR9	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Чтение/запись	Чт. Чт./Зп.	Чт. Чт./Зп.	Чт. Чт./Зп.	Чт. Чт./Зп.	Чт./Зп. Чт./Зп.	Чт./Зп. Чт./Зп.	Чт./Зп. Чт./Зп.	Чт./Зп. Чт./Зп.	
Начальное значение	0 X	0 X	0 X	0 X	X X	X X	X X	X X	

Разряды 15..12 – Резерв

Данные зарезервированные разряды считываются как 0. При записи в данных разрядах необходимо указывать нули для совместимости с новыми версиями микроконтроллеров.

Разряды 11..0 – EEAR11..0: Адрес ячейки ЭСППЗУ

Регистры адреса ЭСППЗУ – EEARH и EEARL – определяют адрес ячейки ЭСППЗУ в 4 кбайтном пространстве. Байтные ячейки ЭСППЗУ адресуются линейно в диапазоне адресов 0…4096. Начальное значение EEAR неопределенное. Необходимое значение адреса должно быть записано до начала доступа к ЭСППЗУ.

Регистр данных ЭСППЗУ –EEDR

Разряд	7	6	5	4	3	2	1	0	
	Ст.							Мл.	EEDR
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	R/W	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряды 7…0 – EEDR7.0: Данные ЭСППЗУ

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

Регистр управления ЭСППЗУ – EECR

Разряд	7	6	5	4	3	2	1	0	
	–	–	–	–	EERIE	EEMWE	EEWE	EERE	EECR
Чтение/запись	Чт.	Чт.	Чт.	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх.знач.	0	0	0	0	0	0	x	0	

Разряды 7…4 – Резерв

Данные разряды у ATmega128 зарезервированы и считываются как 0.

Разряд 3 – EERIE: Разрешение прерывания по готовности ЭСППЗУ

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

Разряд 2 – EEMWE: Главное разрешение записи в ЭСППЗУ

Бит EEMWE разрешает установку бита EEWE, инициирующего запись в ЭСППЗУ. Данные будут записаны в ЭСППЗУ 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 ЭСППЗУ

Сигнал разрешения записи EEWE является стробирующим сигналом записи для ЭСППЗУ. Для записи в ЭСППЗУ после корректной установки адреса и данных необходимо установить бит EEWE. Перед установкой бита EEWE должен быть установлен бит EEWE, иначе запись в ЭСППЗУ не произойдет. При выполнении операции записи в ЭСППЗУ 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 ЭСППЗУ (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.

ЭСППЗУ нельзя программировать во время записи флэш-памяти из ЦПУ. С учетом этого, перед началом новой записи в ЭСППЗУ 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.

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 ЭСППЗУ, прерывается другим доступом к ЭСППЗУ, 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 ЭСППЗУ

Сигнал разрешения чтения из ЭСППЗУ EERE является стробом чтения ЭСППЗУ. После записи корректного адреса в регистр адреса EEAR бит EERE должен быть установлен к лог.1 для запуска механизма чтения ЭСППЗУ. Чтение из ЭСППЗУ выполняется одновременно с инструкцией, поэтому, запрашиваемые данные готовы для считывания сразу по ее завершении. После чтения из ЭСППЗУ 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 ЭСППЗУ, но и изменить регистр адреса EEAR. Во время доступа к ЭСППЗУ используется калиброванный генератор. В таблице 2 приведено типичное время программирования ЭСППЗУ через ЦПУ.

Таблица 2 – Время программирования ЭСППЗУ

Операция	Количество периодов калиброванного RC-генератора (1).	Типичное время программирования
Запись ЭСППЗУ (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 ЭСППЗУ 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 ЭСППЗУ 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 ЭСППЗУ
out EEARH, r18
out EEARL, r17
; Записываем данные (r16)  в регистр данных ЭСППЗУ
out EEDR,r16
; Записывает лог. 1 в EEMWE
sbi EECR,EEMWE
; Запуск записи в ЭСППУ 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 ЭСППЗУ путем установки EEWE */
EECR |= (1<<EEWE);
}
```

В следующих примерах кодов на Си и Ассемблере представлены функции чтения из ЭСППЗУ. 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 ЭСППЗУ путем установки EERE
sbi EECR,EERE
; Считывание данных из регистра данных ЭСППЗУ
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 ЭППЗУ путем установки EERE */
EECR |= (1<<EERE);
/* Возврат данных из регистра данных ЭСППЗУ*/
return EEDR;
}
```

Запись в ЭСППЗУ в режиме выключения (Power Down)

Если микроконтроллер переводится в режим выключения (Power Down) командой sleep в процессе выполнения операции записи в ЭСППЗУ, 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 ЭСППЗУ перед переводом в режим выключения (Power-down).

Меры предотвращения повреждения данных в ЭСППЗУ

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

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

Повреждение данных в ЭСППЗУ 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 ATmega128 показано в разделе “Сводная таблица регистров”.

Все порты ввода-вывода и периферийные устройства в ATmega128 размещены в пространстве ввода-вывода. Доступ ко всем ячейкам ввода-вывода может быть осуществлен с помощью инструкций LD/LDS/LDD и ST/STS/STD 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. ATmega128 является сложным микроконтроллером, для которого 64 адреса, зарезервированных в кодах операций IN и OUT, не достаточно для поддержки всех имеющихся периферийных устройств. Для расширенной области ввода-вывода, которая находится по адресам $60 - $FF в статическом ОЗУ необходимо использовать только инструкции ST/STS/STD и LD/LDS/LDD. Пространство расширенного ввода-вывода заменяется ячейками статического ОЗУ в режиме совместимости с ATmega103.

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

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

Интерфейс внешней памяти

Характеристики интерфейса внешней памяти позволяет его использовать не только для подключения к внешнему статическому ОЗУ или флэш-памяти, но и в качестве интерфейса с внешними периферийными устройствами, например, ЖК-дисплеи, АЦП и ЦАП. Его основными отличительными особенностями являются:

- Возможность задания четырех различных по длительности состояний ожидания, в т.ч. без состояния ожидания.
- Возможность установки различных состояний ожидания для разных секторов внешней памяти (размер сектора конфигурируется).
- Возможность выбора количества задействованных разрядов в старшем адресном байте.
- Устройство запоминания состояния шины для минимизации потребления тока (опционально).

Краткий обзор

После разрешения внешней памяти (XMEM) становится доступным адресное пространство за пределами внутреннего статического ОЗУ через предопределенные для этой функции выводы (см. рис. 1, табл. 27, табл. 33 и табл. 45). Конфигурация памяти показана на рис. 11.

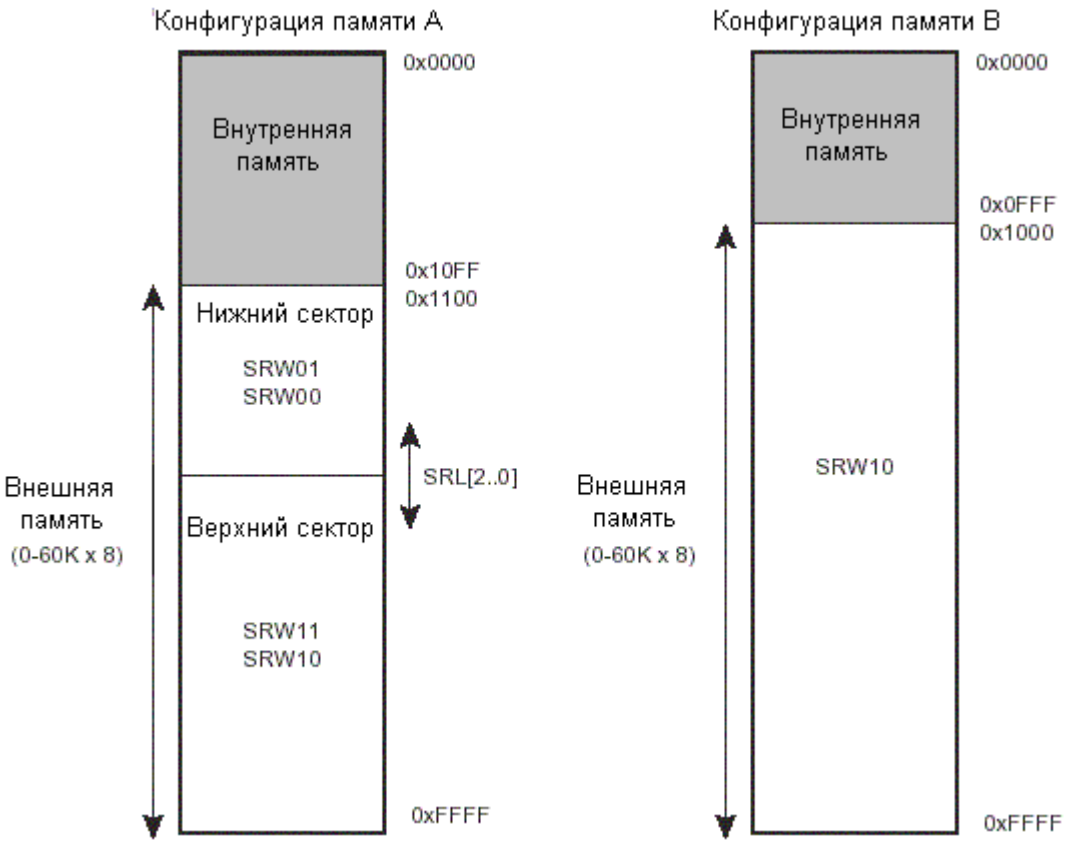


Рисунок 11 – Внешняя память с выбором сектора

Прим. : Когда ATmega128 находится не в режиме совместимости с ATmega103, то доступна конфигурация памяти A (конфигурация B не доступна), а когда ATmega128 в режиме совместимости с ATmega103, то доступна только конфигурация B.

Совместимость с ATmega103

Оба регистра управления внешней памятью (XMCRA и XMCRB) размещены в расширенном пространстве ввода-вывода. В режиме совместимости с ATmega103 эти регистры и функции, управляемые через них, не доступны. Но микроконтроллер сохранит совместимость, т.к. эти функции не поддерживаются у ATmega103. Ограничения на режим совместимости с ATmega103 следующие:

Доступны только две установки состояний ожидания (SRW1n = 0b00 и SRW1n = 0b01).
Количество используемых разрядов в старшем адресном байте является фиксированным.
Внешняя память не может быть поделена на сектора с различными состояниями ожидания.
Устройство запоминания состояния шины не доступно.
Выводы RD, WR и ALE могут использоваться только для вывода (Порт G у ATmega128).

Использование интерфейса внешней памяти

Интерфейс состоит из:

AD7:0: Мультиплексированная младшая шина адреса/шина данных.
A15:8: Старшая шина адреса (с конфигурируемым числом разрядов).
ALE: Строб адреса внешней памяти.
RD: Строб чтения из внешней памяти.
WR: Строб записи во внешнюю память.

Биты управления интерфейсом внешней памяти расположены в трех регистрах: регистр управления микроконтроллером – MCUCR, регистр A управления внешней памятью – XMCRA и регистр B управления внешней памятью – XMCRB.

После разрешения работы интерфейс XMEM изменит настройки регистров направления данных портов, линии которых предопределены для выполнения функций интерфейса XMEM. Более подробная информация об изменении настроек порта приведена в разделе "Порты ввода-вывода" при рассмотрении альтернативных функций. Интерфейс XMEM автоматически определяет к какой памяти внешней или внутренней осуществляется доступ. Во время доступа к внешней памяти интерфейс XMEM будет формировать сигналы шин адреса, данных и управления на линиях порта в соответствии с рисунком 13 (на рисунке представлены формы сигналов без состояний ожидания). При переходе ALE из 1 в 0 на линиях AD7:0 будут присутствовать действительные адресные сигналы. ALE находится на низком уровне во время передачи данных. После разрешения работы интерфейса XMEM доступ к внутренней памяти будет вызывать изменения на шинах данных и адреса, а также строба ALE, при этом, стробы RD и WR останутся неизменными. После запрета работы интерфейса внешней памяти используются обычные установки выводов и направления данных. Обратите внимание, что после отключения интерфейса XMEM адресное пространство свыше внутреннего ОЗУ не связано с последним. Рисунок 12 иллюстрирует как подключить внешнее статическое ОЗУ к AVR-микроконтроллеру с помощью 8-разр. регистра (например, “74x573” или аналогичный), который передает данные напрямую при высоком уровне на входе G.

Требования по фиксации адреса

Интерфейс XRAM характеризуется высоким быстродействием из-за чего фиксация адреса должна выполняться с осторожностью при частотах свыше 8МГц при 4В и 4МГц при 2.7В.

При использовании более высоких частот применение регистров устаревшей серии 74HC будет неадекватным. Интерфейс внешней памяти разработан для совместимости с регистрами серии 74AHC. Однако, большинство регистров может быть использовано, если они отвечают требованиям временной диаграммы. К основным параметрам, характеризующих фиксацию адреса, относятся:

Длительность задержки на распространение сигнала с входа D на выход Q (tPD).
Время установки данных перед тем как G станет равным 0 (tSU).
Время удержания данных (адреса) после установки низкого уровня на входе G (tH).

Интерфейс внешней памяти разработан с учетом того, что после приложения низкого уровня на вход регистра G время удержания адреса tH составит до 5 нс. См. временные характеристики tLAXX_LD/tLLAXX_ST в таблицах 137…144. Задержка распространения с входа D на выход Q (tPD) должна быть учтена при вычислении требования по времени доступа к внешней памяти. Время установки данных перед тем как G станет равным 0 (tSU) должно не превышать разности времен действительного адреса до низкого уровня ALE (tAVLLC) и задержек в печатных проводниках (определяют емкостную нагрузку).

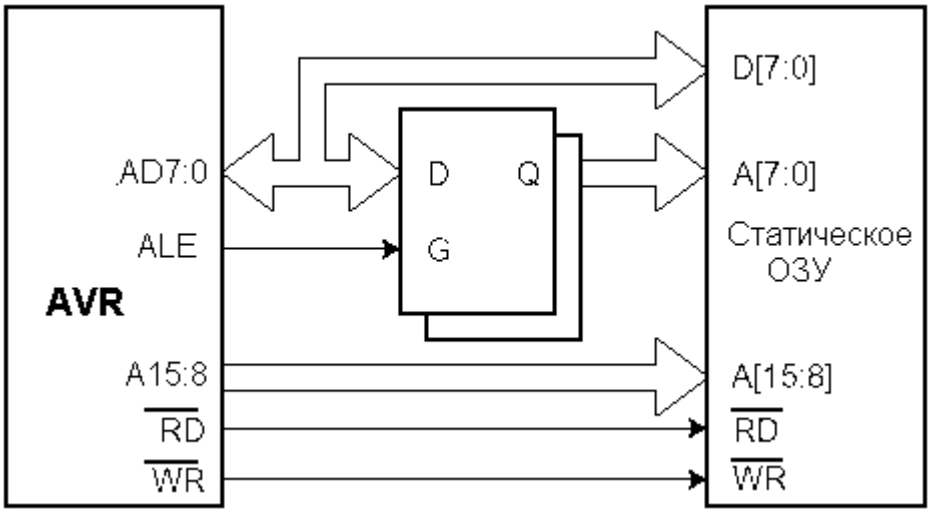


Рисунок 12 – Подключение внешнего статического ОЗУ к AVR-микроконтроллеру

Подтягивающие резисторы и устройство запоминания состояния шины

Подтягивающие к плюсу резисторы на линиях AD7:0 могут быть активизированы, если записать единицы в регистр соответствующего порта. Для снижения потребляемой мощности в режиме сна рекомендуется отключать подтягивающие резисторы путем записи нуля в регистр порта непосредственно перед переводом в режим сна.

Интерфейс XMEM также содержит устройство запоминания состояния шины на линиях AD7:0 Устройство запоминания состояния шины может быть программно подключено и отключено как описано в подразделе “Управляющий регистр B внешней памяти – XMCRB”. После активизации устройство запоминания состояния шины будет сохранять предыдущее состояние шины AD7:0 при переводе этих линий интерфейсом XMEM в третье состояние.

Временная диаграмма

Микросхемы внешней памяти характеризуются различными параметрами временных диаграмм. Для удовлетворения этих требований интерфейс XMEM у ATmega128 обеспечивает различные состояния ожидания (см. табл. 4). Перед выбором состояний ожидания очень важно уточнить требования к временной диаграмме микросхемы внешней памяти. Самыми главными параметрами являются время доступа к внешней памяти по сравнению с требованием по установке у ATmega128. Время доступа к внешней памяти определяется как промежуток времени с момента выбора микросхемы памяти и установки адреса до появления действительных данных соответствующих указанному адресу на шине. Время доступа не может превышать времени с момента установки импульса ALE к низкому уровню до стабильного установления данных во время чтении (см. tLLRL+ tRLRH - tDVRH в таблицах 137…144). Различные состояния ожидания устанавливаются программно. Реализована дополнительная функция, которая позволяет разделить внешнюю память на два сектора и для каждого из них индивидуально выполнить настройку состояний ожидания. Это делает возможным подключить две различных микросхемы памяти с различными требованиями к временной диаграмме доступа через один и тот же интерфейс XMEM. Характеристики временной диаграммы интерфейса XMEM приведены в таблицах 137… 144 и на рисунках 156…159 в разделе “Временная диаграмма внешней памяти данных”.

Обратите внимание, что интерфейс XMEM –асинхронный и что форма сигналов на следующих рисунках связана с внутренней синхронизацией. Расхождение между внутренней и внешней синхронизацией (XTAL1) не гарантируется (зависит от температуры и напряжения питания микросхем). Следовательно, интерфейс XMEM не подходит для синхронной работы.

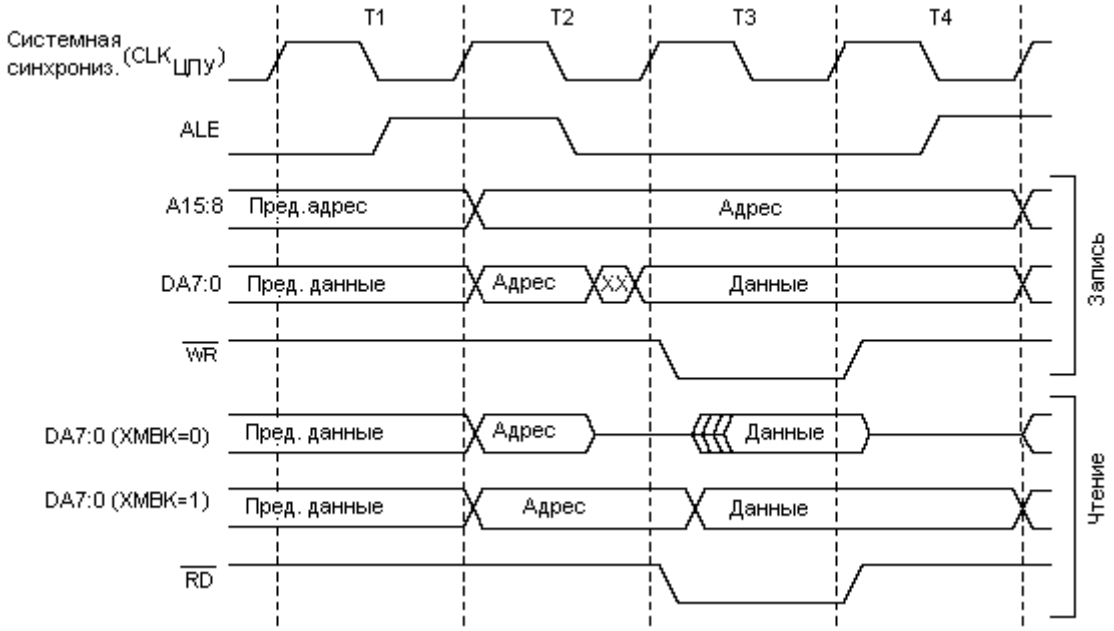


Рисунок 13 – Временная диаграмма доступа к внешней памяти без состояний ожидания (SRWn1=0 и SRWn0=0).

Прим: 1. SRWn1 = SRW11 (верхний сектор) или SRW01 (нижний сектор), SRWn0 = SRW10 (верхний сектор) или SRW00 (нижний сектор). Импульс ALE присутствует на такте T5, только если следующая инструкция осуществляет доступ к ОЗУ (внутреннему или внешнему).

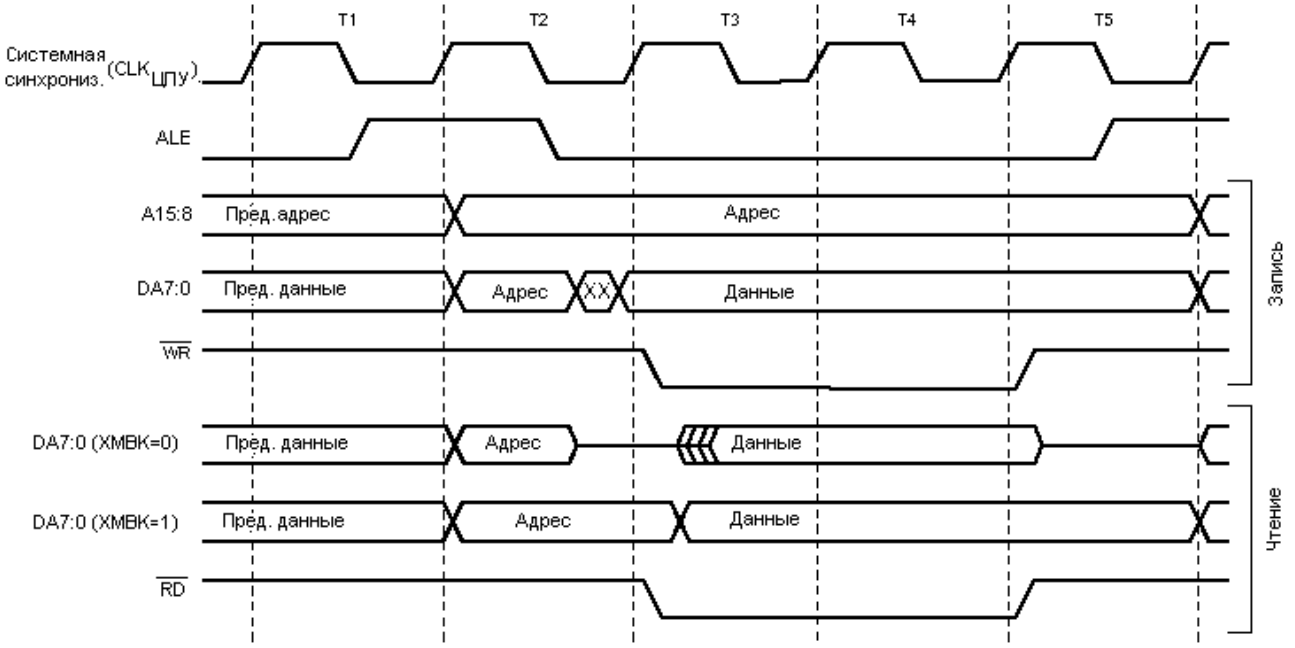


Рисунок 14 – Временная диаграмма доступа к внешней памяти с SRWn1=0 и SRWn0=1 (1).

Прим: 1. SRWn1 = SRW11 (верхний сектор) или SRW01 (нижний сектор), SRWn0 = SRW10 (верхний сектор) или SRW00 (нижний сектор). Импульс ALE присутствует на такте T5 только если следующая инструкция осуществляет доступ к ОЗУ (внутреннему или внешнему).

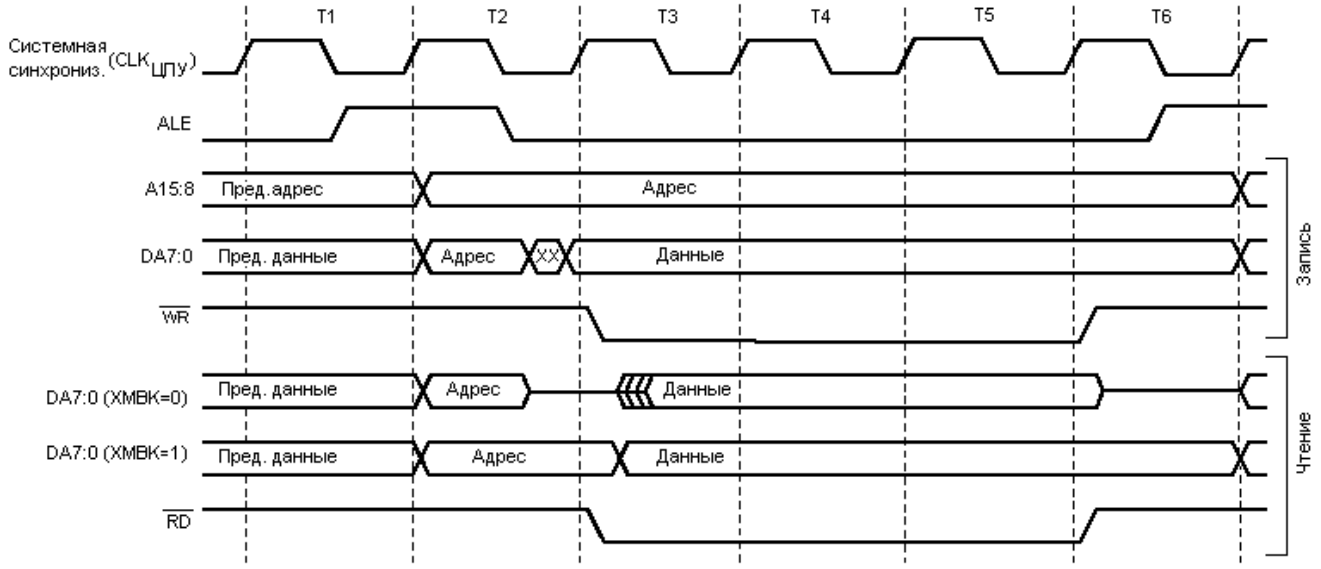


Рисунок 15 – Временная диаграмма доступа к внешней памяти с SRWn1=1 и SRWn0=0 (1).

Прим.: 1. SRWn1 = SRW11 (верхний сектор) или SRW01 (нижний сектор), SRWn0 = SRW10 (верхний сектор) или SRW00 (нижний сектор). Импульс ALE на такте T6 присутствует только если следующая инструкция осуществляет доступ к ОЗУ (внутреннему или внешнему).

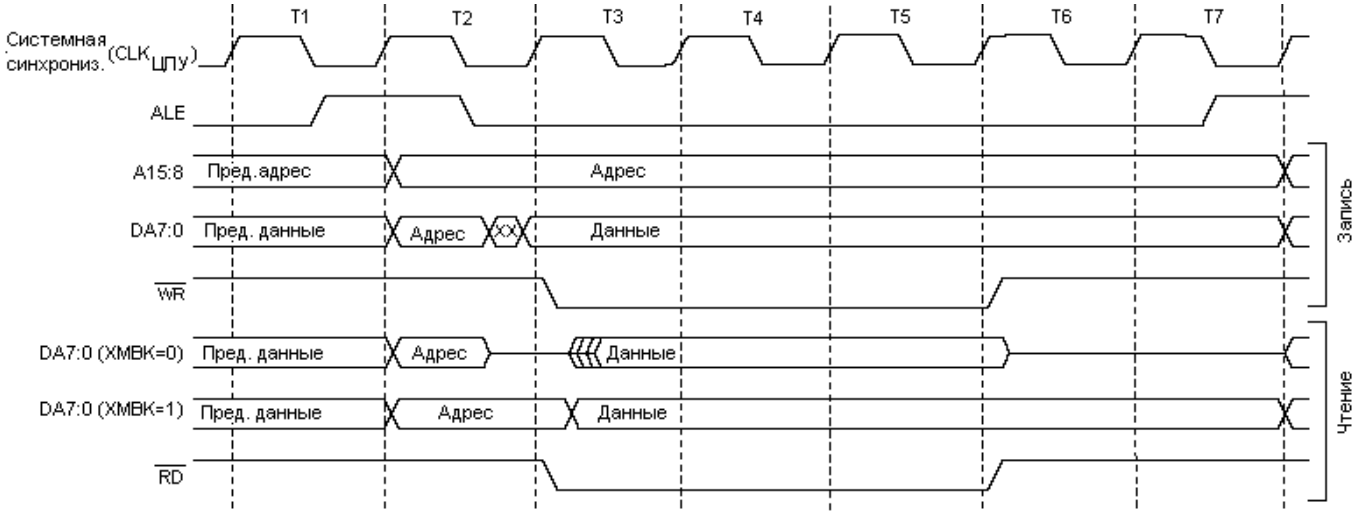


Рисунок 16 – Временная диаграмма доступа к внешней памяти с SRWn1=1 и SRWn0=1 (1).

Прим: 1. SRWn1 = SRW11 (верхний сектор) или SRW01 (нижний сектор), SRWn0 = SRW10 (верхний сектор) или SRW00 (нижний сектор). Импульс ALE на такте T7 присутствует только, если следующая инструкция осуществляет доступ к ОЗУ (внешнему или внутреннему).

Описание регистра XMEM

Регистр управления микроконтроллером– MCUCR

Разряд	7	6	5	4	3	2	1	0	
	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	MCUCR
Чтение/Запись	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	

Разряд 7 – SRE: Разрешение внешнего статического ОЗУ/XMEM

Запись в SRE лог. 1 разрешает работу интерфейса внешней памяти, после чего выводы AD7:0, A15:8, ALE, WR и RD выполняют свои альтернативные функции. После установки бита SRE игнорируются любые установки в соответствующих регистрах направления данных. Запись в SRE нуля отключает интерфейс внешней памяти, после чего вступают в силу обычные функции выводов и установки направления.

Разряд 6 – SRW10: Бит выбора состояния ожидания

При работе микроконтроллера не в режиме совместимости с ATmega103 описание действия данного бита подробно описывается ниже при рассмотрении бит SRWn регистра XMCRA. В режиме совместимости с ATmega103 запись в SRW10 лог. 1 разрешает состояние ожидание и один дополнительный период добавляется к стробу чтения/записи как показано на рис. 14.

Регистр A управления внешней памятью – XMCRA

Разряд	7	6	5	4	3	2	1	0	
	–	SRL2	SRL1	SRL0	SRW01	SRW00	SRW11	–	XMCRA
Чтение/Запись	Чт	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – Зарезервированный бит

Данный разряд является зарезервированным и всегда читается как 0. Во время записи в данной позиции необходимо указывать 0 для совместимости с последующими микроконтроллерами.

Разряд 6..4 – SRL2, SRL1, SRL0: Задание границ секторов с состоянием ожидания

Имеется возможность установить различные состояния ожидания для различных адресов внешней памяти. Адресное пространство внешней памяти может быть разделено на два сектора, каждый их которых имеет собственные биты выбора состояний ожидания. Биты SRL2, SRL1 и SRL0 определяют разбиение секторов (см. табл. 3 и рис. 11). По умолчанию значение бит SRL2, SRL1 и SRL0 равно нулю и все адресное пространство внешней памяти обслуживается как один сектор. Если все адресное пространство статического ОЗУ конфигурируется как один сектор, то состояния ожидания определяются битами SRW11 и SRW10.

Таблица 3 – Границы секторов памяти при различных настройках SRL2..0

SRL2	SRL1	SRL0	Границы сектора
0	0	0	Нижний сектор = Нет Верхний сектор = 0x1100 - 0xFFFF
0	0	1	Нижний сектор = 0x1100 - 0x1FFF Верхний сектор = 0x2000 - 0xFFFF
0	1	0	Нижний сектор = 0x1100 - 0x3FFF Верхний сектор = 0x4000 - 0xFFFF
0	1	1	Нижний сектор = 0x1100 - 0x5FFF Верхний сектор = 0x6000 - 0xFFFF
1	0	0	Нижний сектор = 0x1100 - 0x7FFF Верхний сектор = 0x8000 - 0xFFFF
1	0	1	Нижний сектор = 0x1100 - 0x9FFF Верхний сектор = 0xA000 - 0xFFFF
1	1	0	Нижний сектор = 0x1100 - 0xBFFF Верхний сектор = 0xC000 - 0xFFFF
1	1	1	Нижний сектор = 0x1100 - 0xDFFF Верхний сектор = 0xE000 - 0xFFFF

Разряд 1 и разряд 6 регистра MCUCR – SRW11, SRW10: Биты выбора состояний ожидания для верхнего сектора

Биты SRW11 и SRW10 задают число состояний ожидания для верхнего сектора внешней памяти в соответствии с рисунком 4.

Разряды 3..2 – SRW01, SRW00: Биты выбора состояний ожидания для нижнего сектора

Биты SRW01 и SRW00 задают число состояний ожидания для нижнего сектора внешней памяти (см. табл. 4).

Таблица 4 – Состояния ожидания (1)

SRWn1	SRWn0	Состояния ожидания
0	0	Нет состояний ожидания
0	1	Задержка на один машинный цикл во время строба чтения/записи
1	0	Задержка на два машинных цикла во время строба чтения/записи
1	1	Задержка на два машинных цикла во время строба чтения/записи и задержка на один машинный цикл перед установкой нового адреса

Прим.: 1. n = 0 или 1 для нижнего или верхнего сектора, соответственно. На рисунках 13…16 представлены временные диаграммы, которые соответствуют различным установкам бит SRW.

Разряд 0 – Зарезервированный бит

Данный бит является зарезервированным, поэтому, всегда считывается как ноль. Если осуществляется запись в данную ячейку, то для совместимости с последующими микроконтроллерами рекомендуется в позиции данного бита указывать 0.

Регистр B управления внешней памятью – XMCRB

Разряд	7	6	5	4	3	2	1	0	
	XMBK	–	–	–	–	XMM2	XMM1	XMM0	XMCRB
Чтение/Запись	Чт/Зп	Чт	Чт	Чт	Чт	Чт/Зп	Чт/Зп	Чт/Зп	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7– XMBK: Разрешение работы устройства запоминания состояния шины внешней памяти

Запись в XMBK лог. 1 разрешает работу устройства запоминания состояния шины на линиях AD7:0. После его активизации AD7:0 будут запоминать последнее установленное состояние, даже если интерфейс XMEM перевел линии в третье состояние. Запись в XMBK лог. 0 означает запрет работы устройства запоминания состояния шины. XMBK не подчинен SRE, так что даже если интерфейс XMEM отключен, то устройство запоминания состояния шины будет активным до тех пор пока XMBK=1.

Разряды 6..4 – Зарезервированные разряды

Данные разряды являются зарезервированными для будущих микроконтроллеров, поэтому, для совместимости с ними рекомендуется записывать в данные позиции лог. 0 во время записи в данный регистр.

Разряды 2..0 – XMM2, XMM1, XMM0: Маска старших адресных разрядов внешней памяти

После разрешения внешней памяти все выводы порта C по умолчанию используются в качестве старшего адресного байта. Если нет необходимости адресоваться ко всему 60 кбайтному

пространству внешней памяти, то свободные адресные линии возможно использовать в качестве универсального ввода-вывода (см. табл. 5). Использование бит XMMn позволяет адресоваться ко всем 64 кбайт ячейкам внешней памяти (см. “Использование всех 64 кбайт ячеек внешней памяти".

Таблица 5 – Использование старших адресных сигналов в качестве линий универсального ввода-вывода после разрешения внешней памяти

XMM2	XMM1	XMM0	Число разрядов адреса внешней памяти	Освобождаемые адресные линии порта C
0	0	0	8 (Все пространство 60 кбайт)	Нет
0	0	1	7	PC7
0	1	0	6	PC7 - PC6
0	1	1	5	PC7 - PC5
1	0	0	4	PC7 - PC4
1	0	1	3	PC7 - PC3
1	1	0	2	PC7 - PC2
1	1	1	Старший байт адреса не используется	Полностью порт C

Использование всех ячеек внешней памяти размером менее 64 кбайт

Поскольку в соответствии с рисунком 11 адресное пространство внешней памяти следует за адресным пространством внутренней, то к младшим 4352 ячейкам внешней памяти не возможно адресоваться (адреса 0x0000…0x10FF). Однако, при подключении внешней памяти размером менее 64 кбайт, например, 32 кбайт к этим ячейкам можно легко адресоваться по адресам 0x8000…0x90FF. Поскольку адресный бит внешней памяти A15 не подключен к внешней памяти, то адреса 0x8000…0x90FF будут выступать в качестве адресов 0x0000…0x10FF для внешней памяти. Адресация по адресам свыше 0x90FF не рекомендуется, т.к. может затронуть ячейку внешней памяти, доступ к которой уже осуществлялся по другому (меньшему) адресу. Для прикладной программы 32 кбайта внешней памяти будут представлять линейное адресное пространство с адресами 0x1100…0x90FF (см. рис. 17). Конфигурация памяти B относится к режиму совместимости с ATmega103, а конфигурация памяти A к нормальному режиму работы.

Когда микроконтроллер находится в режиме совместимости с ATmega103 внутренняя память занимает 4096 байт. Это означает, что 4096 байт внешней памяти могут быть доступны по адресам 0x8000…0x8FFF. Для прикладной программы внешняя память размером 32 кбайт в этом случае будет линейным адресным пространством в диапазоне адресов 0x1000…0x8FFF.

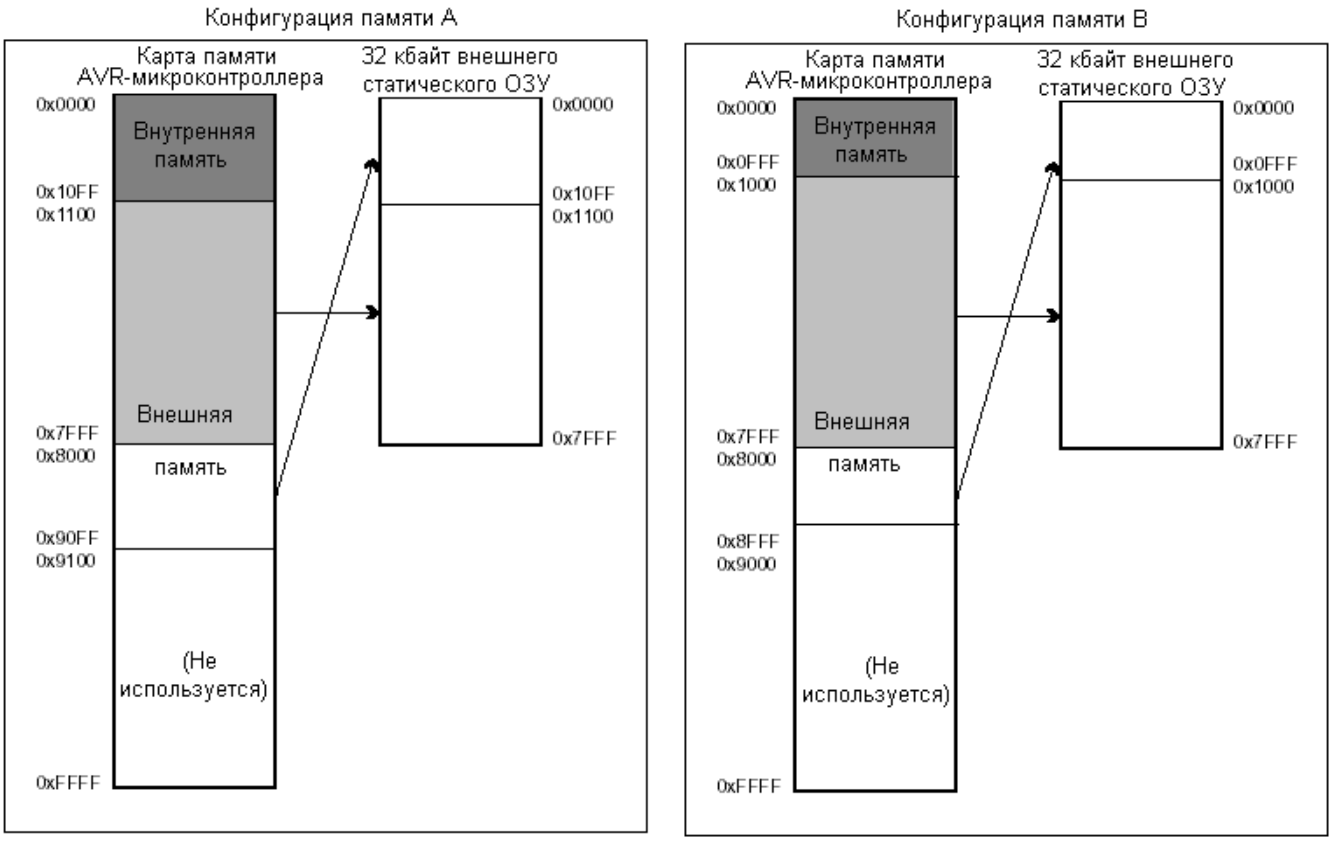


Рисунок 17 – Подключение 32 кбайт внешней памяти

Использование всех 64 кбайт ячеек внешней памяти

Поскольку, внешняя память располагается после внутренней памяти, как показано на рис. 11, то только 60 кбайт внешней памяти доступно по умолчанию (адресное пространство от 0x0000 до 0x10FF зарезервировано для внутренней памяти). Однако, имеется возможность использовать весь объем внешней памяти путем маскирования старших адресных разрядов к нулю. Это может быть выполнено с помощью бит XMMn и программного управления старшими адресными разрядами. Интерфейс памяти будет иметь диапазон адресов 0x0000 - 0x1FFF, если установить на выходе порта C значение 0x00 и выбрать работу старших адресных разрядов как обычных линий ввода-вывода. См. нижеприведенные примеры.

Пример кода на Ассемблере (1)

```
; Определено СМЕЩЕНИЕ OFFSET=0x2000 для гарантирования доступа к внешней
памяти
; Конфигурируем порт C (старший байт адреса ) на вывод 0x00, после чего
настраиваем выводы
; на выполнение функций обычного порта
ldi r16, 0xFF
out DDRC, r16
ldi r16, 0x00
out PORTC, r16
; освобождаем PC7:5 от адресных функций
ldi r16, (1<<XMM1)|(1<<XMM0)
sts XMCRB, r16
; запись  0xAA по адресу 0x0001 внешней памяти
ldi r16, 0xaa
sts 0x0001+OFFSET, r16
; Разрешаем снова работу  PC7:5  как адресных линий
ldi r16, (0<<XMM1)|(0<<XMM0)
sts XMCRB, r16
; Запись 0x55 по адресу (OFFSET + 1) внешней памяти
ldi r16, 0x55
```

```
    sts 0x0001+OFFSET, r16
```

Пример кода на Си (1)

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

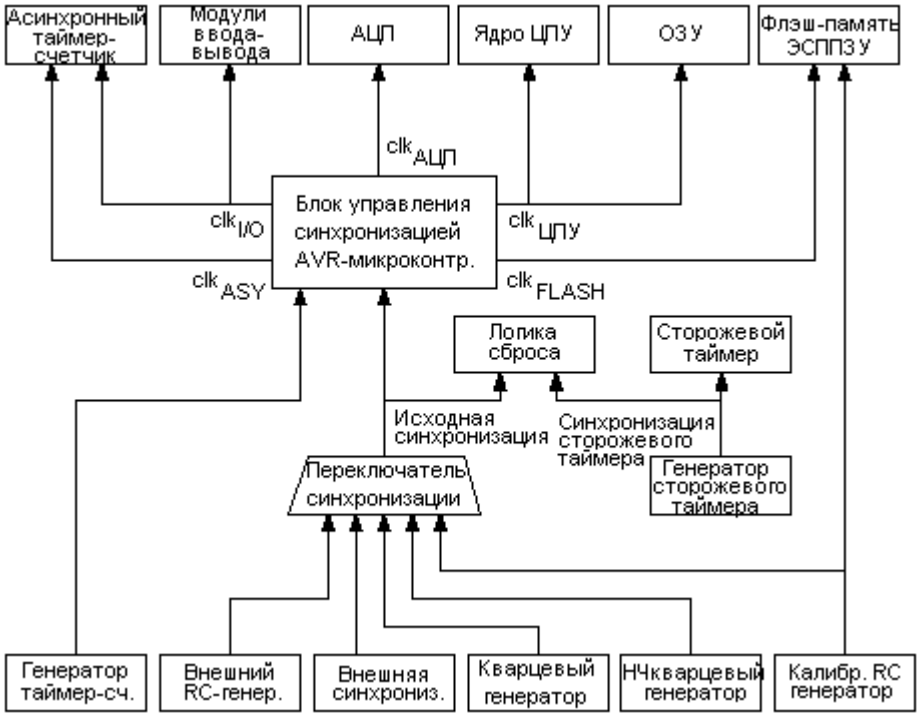
Прим.: 1. В примере предполагается, что включена часть специфического файла заголовка.

Системная синхронизация и тактовые источники

Источники синхронизации и их распределение

На рисунке 18 представлены источники синхронизации и распределение синхроимпульсов к встроенным блокам ATmega128. 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 18 – 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.

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 6 – 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 7. Частота генератора сторожевого таймера зависит от напряжения питания, что показано в разделе “Типовые характеристики ATmega128: предварительные данные”.

Таблица 7 – 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)

Первоначальный источник синхронизации

Микроконтроллер поставляется с установками CKSEL = “0001” и SUT = “10”. Эти значения соответствуют выбору в качестве источника синхронизации внутреннего RC-генератора с максимальным временем старта. Данная настройка гарантирует всем пользователям возможность установить требуемый источник синхронизации с помощью внутрисистемного или параллельного программатора.

Кварцевый генератор

XTAL1 и XTAL2 – вход и выход, соответственно, инвертирующего усилителя, который может быть настроен для использования в качестве встроенного генератора (см. рисунок 19). 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.

Таблица 8- Рабочие режимы кварцевого генератора

CKOPT	CKSEL3..1	Частотный диапазон(1), МГц	Рекомендуемый диапазон номиналов C1 и C2 при использовании кварцевого резонатора
1	101 (2)	0.4-0.9	-
1	110	0.9-3.0	12пФ-22пФ
1	111	3.0-8.0	12пФ-22пФ
0	101, 110, 111	1.0-	12пФ-22пФ

Примечания:

1. Частотные диапазоны – ориентировочные данные. Фактические значения могут отличаться.
2. 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 9.

Таблица 9 – Временая 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.:

1. 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.
2. 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 19. 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 10.

Таблица 10 – 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		

Примечание:

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 20. 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. Более подробная информация о работе генератора и о выборе номиналов R и C приведена в рекомендациях по применению внешнего RC-генератора.

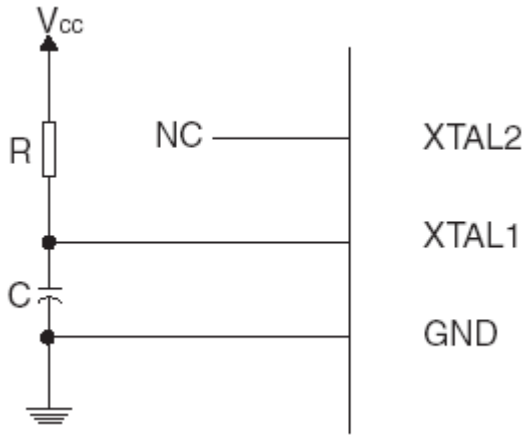


Рисунок 20 – Схема подключения внешней задающей RC-цепи

Генератор может работать в четырех различных режимах, каждый из которых ориентирован на специфический частотный диапазон. Рабочий режим выбирается конфигурационными битами CKSEL3..0 (см. табл. 11).

Таблица 11 – 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. 12).

Таблица 12 – 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

Примечание:

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 13. 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 13 – 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. 14). Выводы XTAL1 и XTAL2 должны быть оставлены неподключенными (NC).

Таблица 14 – 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

Прим.: Регистр OSCCAL недоступен в режиме совместимости с ATmega103.

Разряд	7	6	5	4	3	2	1	0	
	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Начальное значение	*Калибровочное значение индивидуально для каждого микроконтроллера*								

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 ЭСППЗУ. 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 ЭСППЗУ и флэш-памяти. Во время выполнения записи в ЭССПЗУ или во флэш-память не следует выполнять калибровку на частоту выше на 10% от номинальной. В противном случае, запись в ЭССПЗУ 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 15.

Таблица 15 – 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.

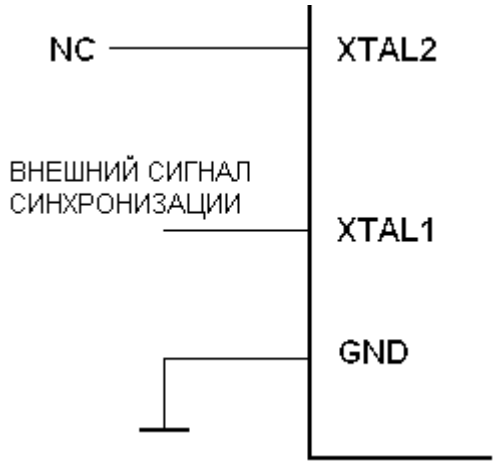


Рисунок 21 – Схема подключения внешнего источника синхронизации

После выбора данного источника синхронизации длительность задержки при запуске определяется конфигурационными битами SUT как показано в таблице 16.

Таблица 16 – 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.

Регистр управления делением XTAL – XDIV

Регистр управления делением XTAL используется для деления частоты тактового источника на одно из значений в диапазоне 2 - 129. Данная функция может использоваться при необходимости оптимизации энергопотребления.

Разряд	7	6	5	4	3	2	1	0
	XDIVEN	XDIV6	XDIV5	XDIV4	XDIV3	XDIV2	XDIV1	XDIV0
Чтение/Запись	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

Разряд 7 – XDIVEN: Разрешение деления XTAL

Если в XDIVEN записать лог.1, то тактовые частоты ЦПУ и периферийных модулей (clkI/O, clkADC, clkCPU, clkFLASH) будут поделены на коэффициент, заданный значениями XDIV6 - XDIV0. Данный бит можно программировать в работающем приложении для варьирования тактовой частотой.

Разряды 6..0 – XDIV6..XDIV0: Разряды 6…0 коэффициента деления XTAL

Данные разряды определяют значение коэффициента деления, который вступает в силу после записи лог. 1 в XDIVEN. Если значение данных бит обозначит как d, то результирующая тактовая частота (fCLK) ЦПУ и периферийных модулей может быть найдена по выражению:

$$f_{CLK} = \frac{\text{Частота тактового источника}}{129\text{-}d}$$

Изменять значение данных разрядов допускается, только если XDIVEN=0. Когда в XDIVEN записывается лог.1, то записываемые одновременно с ней значения XDIV6..XDIV0 будут использоваться как коэффициент деления. Во время записи в XDIVEN лог. 0 одновременно записываемые значения в разряды XDIV6..XDIV0 отклоняются. Поскольку, делитель выполняет снижение входной тактовой частоты микроконтроллера, то после разрешения деления также снижается быстродействие всех периферийных модулей.

Примечание: После разрешения деления тактовой частоты таймер-счетчик 0 может быть использован только в асинхронном режиме. Частота асинхронного источника должна быть не менее чем в 4 раза меньше результирующей (поделенной) частоты синхронизации. В противном случае не гарантируется определение запроса на прерывание и корректность доступа к регистрам таймера-счетчика 0.

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 шести 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" или расширенный дежурный "Extended Standby") после выполнения команды SLEEP (см. табл. 17). 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.

На рисунке 18 представлены различные системы синхронизации микроконтроллера ATmega128 и их распределение. Данный рисунок может быть полезным при выборе соответствующего режима сна.

Регистр управления микроконтроллером – MCUCR

Регистр управления микроконтроллером содержит биты управления энергопотреблением.

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 17.

Таблица 17 – 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)
1	1	1	Расширенный 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 INT7:4 или внешнем прерывании по входам INT3:0.

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 INT7:4 или внешним прерывании INT3:0. 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 некоторыми конфигурационными битами CKSEL, которые определяют период задержки при сбросе (см. “Источники синхронизации”).

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.

Расширенный дежурный режим (Extended Standby)

Запись в SM2..0 значения 111 с учетом выбора в качестве тактового источника внешнего кварцевого или керамического резонатора означает, что после выполнения команды SLEEP микроконтроллер будет переведен в расширенный дежурный режим. Данный режим идентичен экономичному за исключением продолжения работы тактового генератора.

Выход из расширенного дежурного режима происходит за шесть машинных циклов.

Таблица 18 – Активные тактируемые модули и источники пробуждения в различных режимах сна

Наименование режима сна	Тактируемые модули микроконтроллера					Активные генераторы		Источник пробуждения		
	clkCPU (ЦПУ)	clkFLASH (флэш-амять)	clkIO (ввод-вывод)	clkADC (АЦП)	clkASY (Асинхр.модули)	Основной тактовый	Генератор таймера	INT7:0	Набл. адреса TWI	Таймер 0
Холостой ход			*	*	*	*	*	*	*	*
Уменьшение шумов АЦП				*	*	*	*	*	*	*
Выключение								*	*	
Экономичный					*		*	*	*	*
Дежурный (1)						*		*	*	
Расширенный дежурный (1)					*	*	*	*	*	*

Прим.

1. В качестве внешнего тактового источника выбран кварцевый или керамический резонатор.
2. Если установлен бит AS0 в ASSR.
3. Только INT3:0 или прерывание по уровню на INT7:4

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.

Интерфейс JTAG и встроенный блок отладки

Если работа встроенного блока отладки разрешена конфигурационным битом OCDEN, то даже при переводе микроконтроллера в экономичный режим (Power save) или режим выключения (Power down) командой sleep основной тактовый источник продолжит работу. В этом случае микроконтроллер будет потреблять существенный ток даже в этих режимах сна. Избежать этого можно с помощью одного из трех способов:

Сбросить конфигурационный бит OCDEN.
Сбросить конфигурационный бит JTAGEN.
Установить бит JTD в регистре MCUCSR.

После разрешения работы интерфейса JTAG вывод TDO остается плавающим до тех пор пока JTAG TAP-контроллер не начнет сдвигать данные. Если связанная с выводом TDO аппаратная

часть не выполняет подтягивание потенциала к плюсу питания, то потребляемая мощность увеличится. Обратите внимание, что вывод TDI следующего микроконтроллера в сканируемой цепи содержит подтягивающий резистор для избежания данной проблемы. Запись в бит JTD регистра MCUCSR лог. 1 приводит к отключению интерфейса JTAG, так же как и незапрограммированное состояние конфигурационного бита JTAGEN.

Системное управление и сброс

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 22 показана схема организации логики сброса. В таблице 19 приведены электрические характеристики схемы сброса.

Порты ввода-вывода AVR-микроконтроллера немедленно возвращаются к их первоначальному состоянию, как только один из источников сброса становится активным. Для этого не требуется работа какой-либо синхронизации.

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”.

Источники сброса9

ATmega128 имеет пять источников 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 истек 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.
Сброс через интерфейс JTAG. Микроконтроллер находится в состоянии сброса до тех пор, пока в регистре сброса записана лог. 1 в одной из сканируемых цепей JTAG-системы. См. раздел “Граничное сканирование IEEE 1149.1 (JTAG)”.

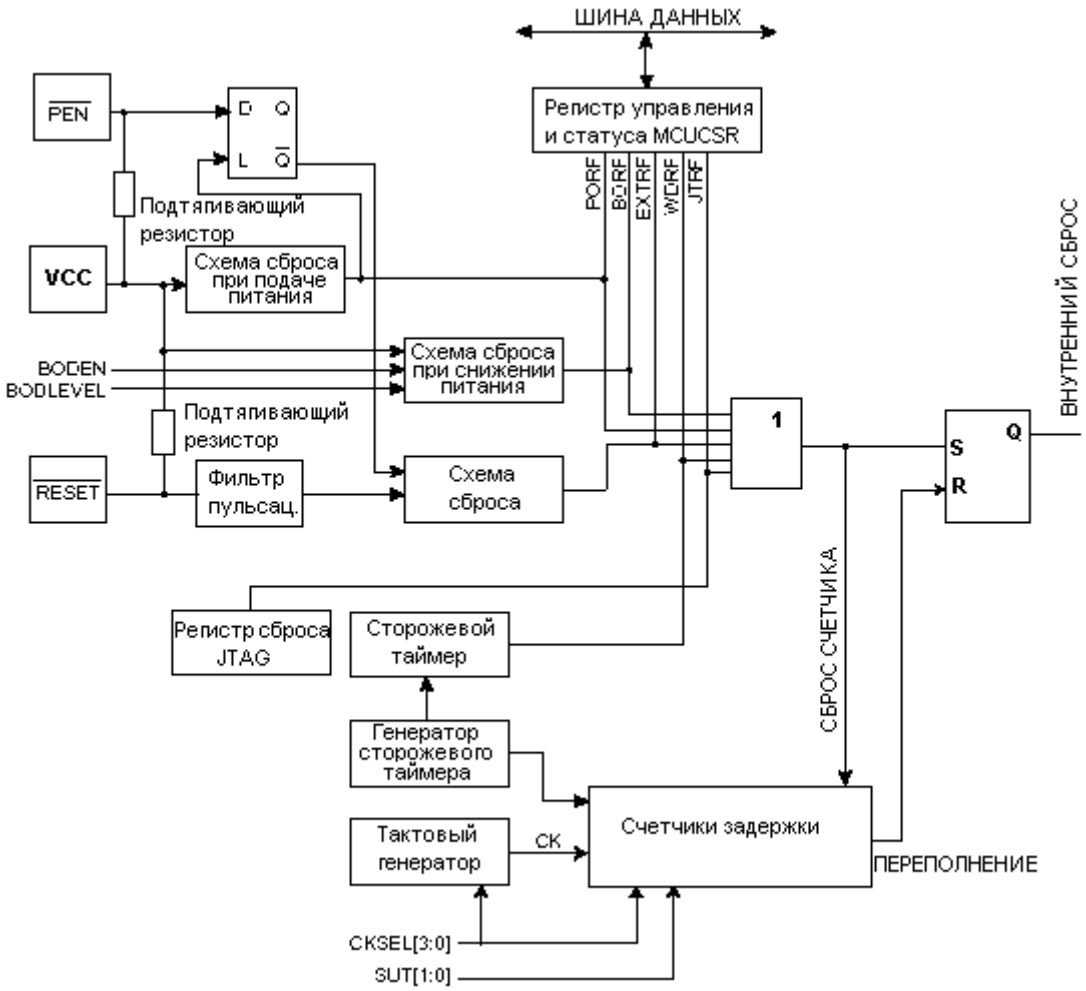


Рисунок 22 – Логика сброса

Таблица 19 – 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		0.85 Vcc	В
tRST	Минимальная длительность импульса сброса на выводе RESET			50		нс
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	В
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			50		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 ATmega128L выполнено при BODLEVEL=1, а ATmega128 при BODLEVEL=0. Установка BODLEVEL=1 не применима для ATmega128.

Сброс при подаче питания

Импульс сброса при подаче питания (POR) генерируется встроенной схемой. Пороговый уровень сброса приведен в таблице 19. POR инициируется всякий раз, когда VCC ниже порогового уровня. Схема POR может использоваться для запуска микроконтроллера, а также для выявления нарушения режима питания.

Функцией схемы сброса при подаче питания (POR) является удержание микроконтроллера в состоянии сброса в течение определенного времени после того, как напряжение достигло уровня сброса при подаче питания. Если напряжение питания снизится ниже определенного уровня, то микроконтроллер снова сбросится без всяких задержек.

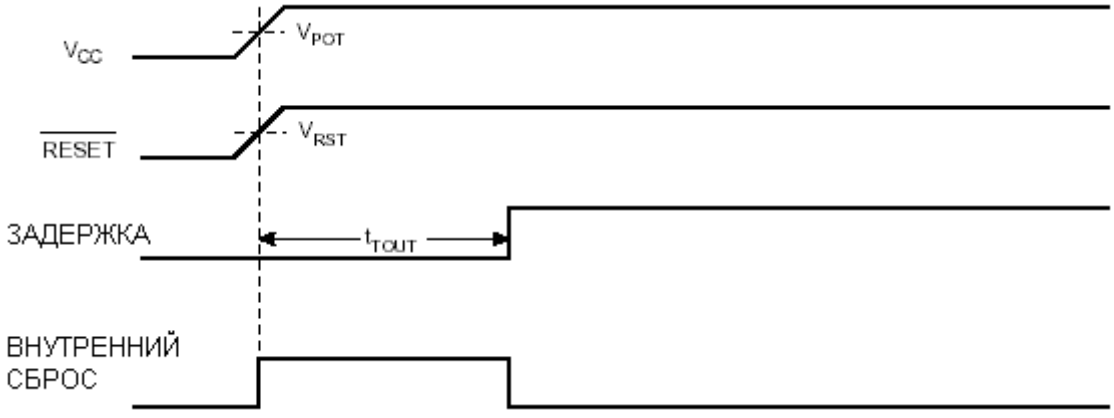


Рисунок 23 – Запуск микроконтроллера (RESET соединен с VCC).

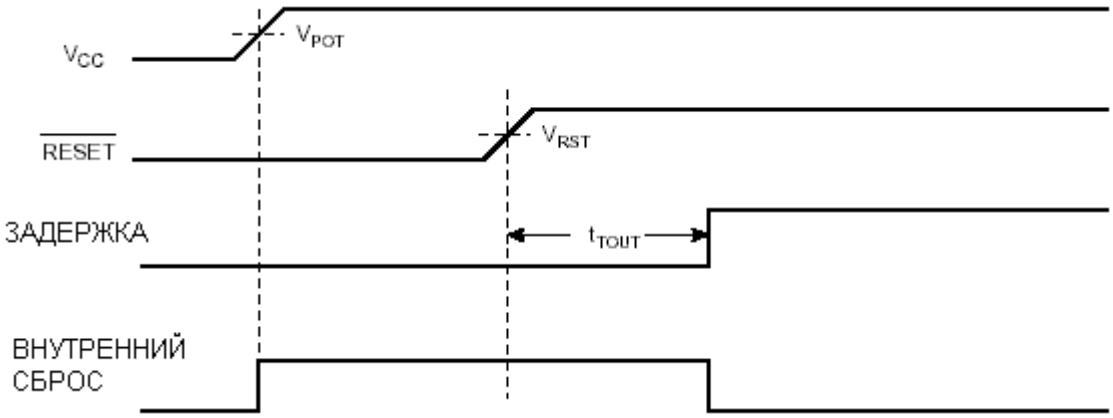


Рисунок 24 – Запуск микроконтроллера (RESET управляется внешне).

Внешний сброс

Внешний сброс генерируется, если на вход RESET подать низкий уровень. Если подать импульс сброса длительностью более tRST (см. табл. 19), 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.

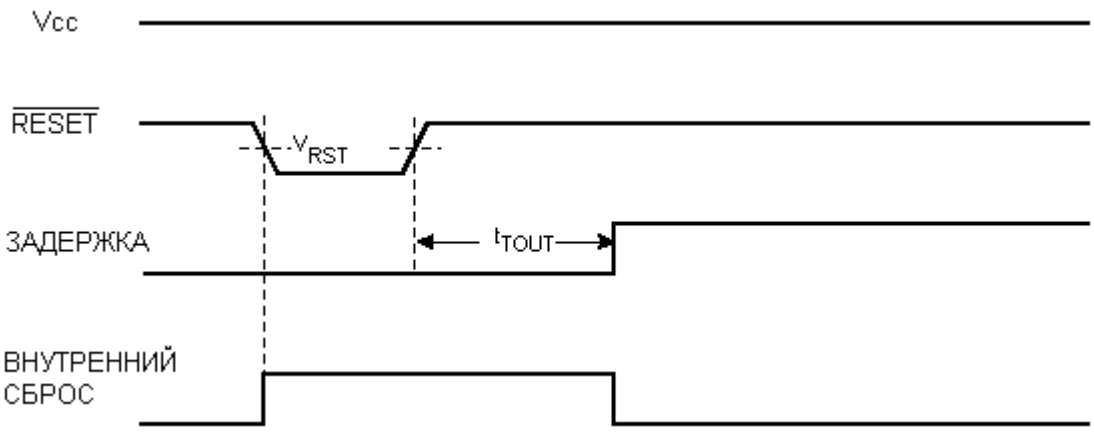


Рисунок 25 – Внешний сброс микроконтроллера

Контроль напряжения питания

ATmega128 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 26), то схема BOD переводит микроконтроллер в состояние сброса. Когда VCC достигает значения выше порога срабатывания (VBOT+ на рисунке 26), 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. 19).

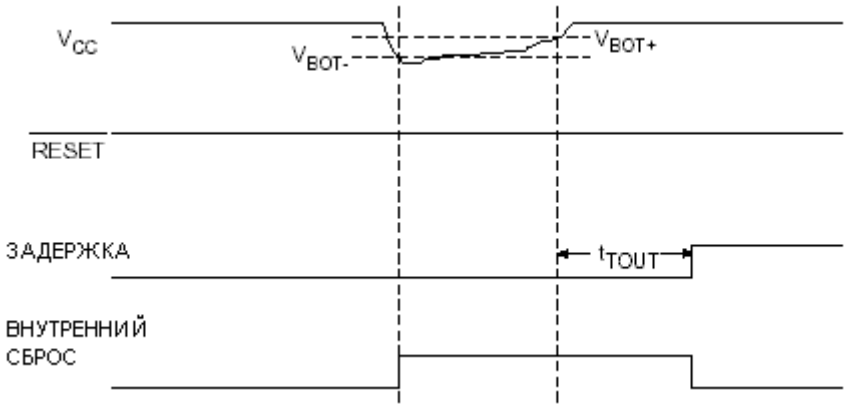


Рисунок 26 – Сброс микроконтроллера схемой контроля питания

Сторожевой таймер

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.

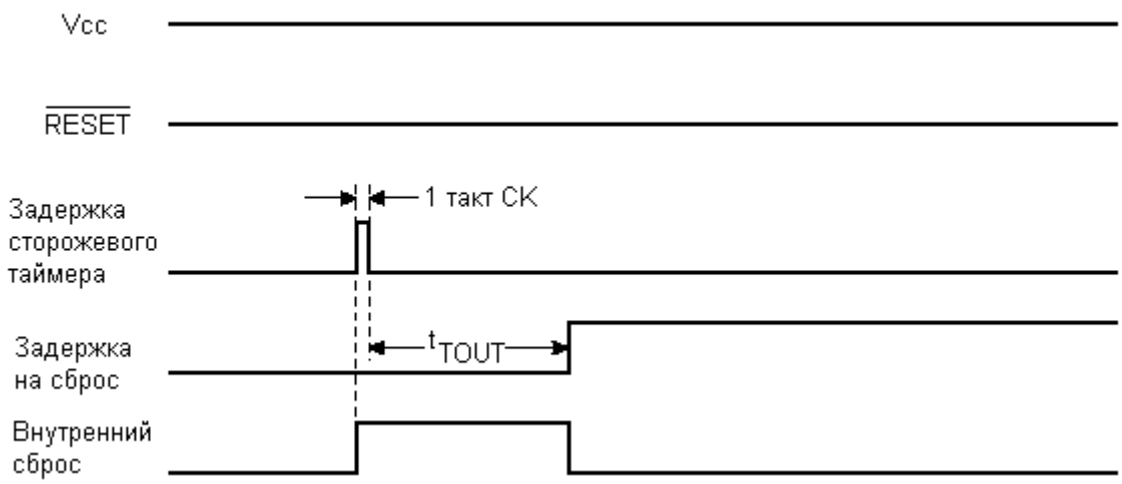


Рисунок 27 – Сброс сторожевым таймером

Регистр управления и статуса микроконтроллера – MCUCSR

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

Разряд	7	6	5	4	3	2	1	0	
	JTD	–	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Чтение/Запись	Чт./Зп.	Чт.	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Нач.значение	0	0	0	См. описание разрядов					

Обратите внимание, что в режиме совместимости с ATmega103 доступны только биты EXTRF и PORF.

Разряд 4 – JTRF: Флаг индикации сброса через JTAG-интерфейс

Данный бит принимает единичное состояние, если сброс был вызван записью лог. 1 в регистр JTAG Reset JTAG-инструкцией AVR_RESET. Данный бит сбрасывается автоматически при подаче питания или путем непосредственной записи лог. 0 в данный флаг.

Разряд 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.

Разряд 2 – BORF: Флаг индикации сброса схемой контроля напряжения питания

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.

Разряд 1 – EXTRF: Флаг внешнего сброса

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

Разряд 0 – PORF: Флаг сброса при подаче 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

ATmega128 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 20 – 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. Значение частоты при другом напряжении питания см. на рис. 194. Период переполнения сторожевого таймера можно задавать путем управления предделителем (см. табл. 22). 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 ATmega128 сбрасывается и начинает выполнение программы по вектору сброса.

Для предотвращения неумышленного изменения периода переполнения или выключения сторожевого таймера могут быть использованы 3 различных уровня безопасности, которые выбираются конфигурационными битами M103C и WDTON в соотв. с табл. 21. Уровень безопасности 0 соответствует установкам в ATmega103. Для разрешения работы сторожевого таймера не требуются какие-либо специальные меры независимо от уровня безопасности (см. “Временные последовательности изменения конфигурации сторожевого таймера ”).

Таблица 21 – Конфигурация сторожевого таймера при различных настройках бит M103C и WDTON

M103C	WDTON	Уровень безопасности	Начальное состояние сторожевого таймера	Временная последовательность отключения сторожевого таймера	Временная последовательность изменения периода переполнения
Незапрограммирован	Незапрограммирован	1	Выкл.	Есть	Есть
Незапрограммирован	Запрограммирован	2	Вкл.	Отключение невозможно	Есть
Запрограммирован	Незапрограммирован	0	Выкл.	Есть	Нет
Запрограммирован	Запрограммирован	2	Вкл.	Отключение невозможно	Есть

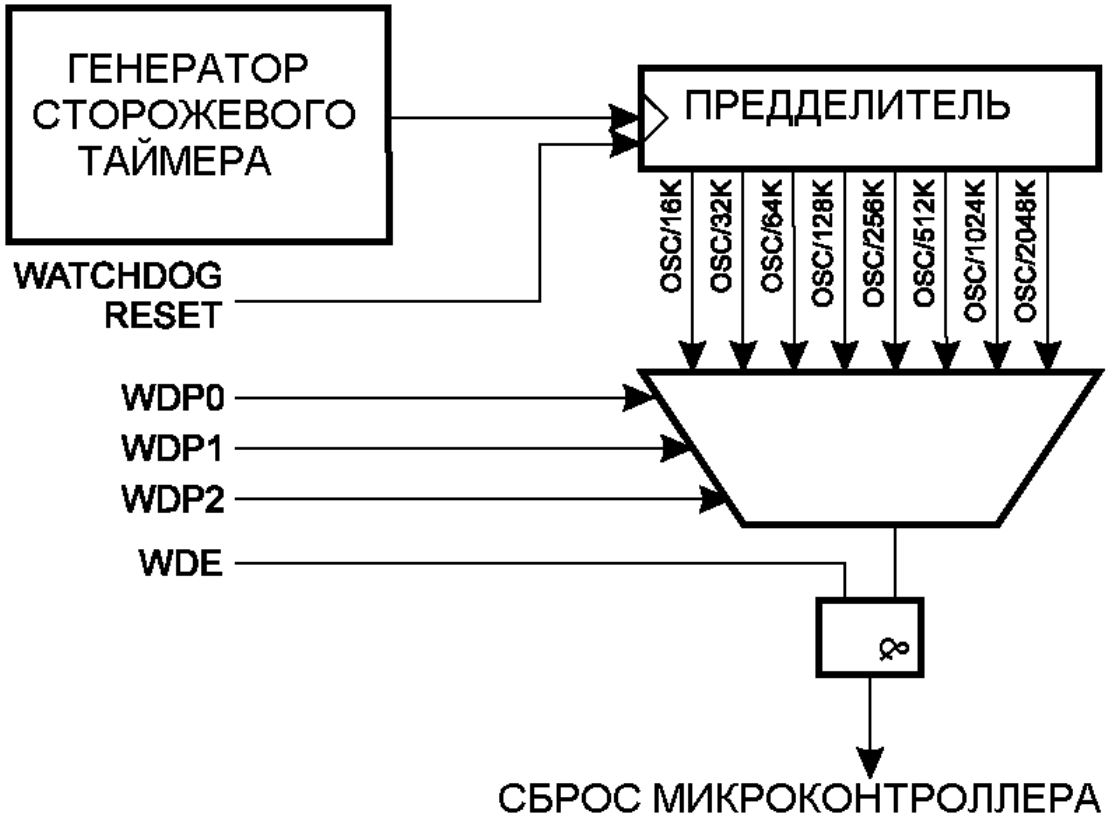


Рисунок 28- Сторожевой таймер

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	

Разряды 7..5 – Зарезервированы

Данные разряды являются зарезервированными и всегда считываются как 0.

Разряд 4 – 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”.

Разряд 3 – 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.

На 2-ом уровне безопасности запретить работу сторожевого таймера невозможно даже с помощью указанной выше последовательности. См. “Временные последовательности изменения конфигурации сторожевого таймера”.

Разряд 2..0 – 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. 22.

Таблица 22 – 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).

Пример кода на Ассемблере

```
WDT_off:
; Запись лог. 1 в WDCE и WDE
ldi 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)
    {
    /* Запись лог. 1 в WDCE и WDE */
    WDTCR = (1<<WDCE) | (1<<WDE);
    /* Выкл. сторожевого таймера */
    WDTCR = 0x00;
    }
```

Временные последовательности изменения конфигурации сторожевого таймера

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

Уровень безопасности 0

Данный режим совместим с работой сторожевого таймера ATmega103. Сторожевой таймер первоначально отключен, но может быть активизирован путем записи в бит WDE лог. 1 без каких-либо ограничений. Период переполнения таймера также может быть изменен без всяких ограничений. Для выключения активизированного сторожевого таймера должна быть выполнена процедура, описанная при рассмотрении бита WDE.

Уровень безопасности 1

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 безопасности 2

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 ATmega128. Общее описание обработки прерываний приведено в разделе “Сброс и обработка прерываний”.

Векторы прерываний в ATmega128

Таблица 23 – Векторы сброса и прерываний

№ вектора	Адрес памяти программ(4)	Источник	Условие возникновения прерывания
1	$0000(1)	RESET	Внешний сброс, сброс при подаче питания, сброс при недопустимом снижении питания, сброс сторожевым таймером и сброс через JTAG-интерфейс
2	$0002	INT0	Запрос на внешнее прерывание 0
3	$0004	INT1	Запрос на внешнее прерывание 1
4	$0006	INT2	Запрос на внешнее прерывание 2
5	$0008	INT3	Запрос на внешнее прерывание 3
6	$000A	INT4	Запрос на внешнее прерывание 4
7	$000C	INT5	Запрос на внешнее прерывание 5
8	$000E	INT6	Запрос на внешнее прерывание 6
9	$0010	INT7	Запрос на внешнее прерывание 7
10	$0012	TIMER2 COMP	Срабатывание компаратора таймера-счетчика 2
11	$0014	TIMER2 OVF	Переполнение таймера-счетчика 2
12	$0016	TIMER1 CAPT	Захват фронта таймером-счетчиком 1
13	$0018	TIMER1 COMPA	Срабатывание компаратора A таймера-счетчика 1
14	$001A	TIMER1 COMPB	Срабатывание компаратора B таймера-счетчика 1
15	$001C	TIMER1 OVF	Переполнение таймера-счетчика 1
16	$001E	TIMER0 COMP	Срабатывание компаратора таймера-счетчика 0
17	$0020	TIMER0 OVF	Переполнение таймера-счетчика 0
18	$0022	SPI, STC	Завершение последовательной передачи интерфейсом SPI
19	$0024	USART0, RX	Завершение приема УСАПП 0
20	$0026	USART0, UDRE	Регистр данных УСАПП0 свободен
21	$0028	USART0, TX	Завершение передачи УСАПП 0
22	$002A	ADC	Завершение преобразования АЦП
23	$002C	EE READY	Готовность ЭСППЗУ
24	$002E	ANALOG COMP	Аналоговый компаратор
25	$0030(3)	TIMER1 COMPC	Срабатывание компаратора C таймера-счетчика 1
26	$0032(3)	TIMER3 CAPT	Захват фронта таймером счетчиком 3
27	$0034(3)	TIMER3 COMPA	Срабатывание компаратора A таймера-счетчика 3
28	$0036(3)	TIMER3 COMPB	Срабатывание компаратора B таймера-счетчика 3

29	$0038(3)	TIMER3 COMPC	Срабатывание компаратора С таймера-счетчика 3
30	$003A(3)	TIMER3 OVF	Переполнение таймера счетчика 3
31	$003C(3)	USART1, RX	Завершение приема УСАПП 1
32	$003E(3)	USART1, UDRE	Регистр данных УСАПП1 свободен
33	$0040(3)	USART1, TX	Завершение передачи УСАПП1
34	$0042(3)	TWI	Двухпроводной последовательный интерфейс
35	$0044(3)	SPM READY	Готовность записи в память 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.
3. Прерывания по адресам $0030 - $0044 не существуют в режиме совместимости с ATmega103.

В таблице 24 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 24 – Размещение векторов сброса и прерываний

BOOTRST	IVSEL	Адрес сброса	Начальный адрес векторов прерываний
1	0	$0000	$0002
1	1	$0000	Адрес сброса в загрузочном секторе + $0002
0	0	Адрес сброса в загрузочном секторе	$0002
0	1	Адрес сброса в загрузочном секторе	Адрес сброса в загрузочном секторе + $0002

Прим. : Адрес сброса загрузочного сектора показан в таблице 113. Для конфигурационного бита BOOTRST “1” означает незапрограммированное состояние, “0” - запрограммированное.

Ниже приведено большинство типичных и общих программных установок адресов сброса и векторов прерываний у ATmega128:

```
Адрес   Инструкция       Комментарий
$0000   jmp RESET        ; Переход на обработку сброса
$0002   jmp EXT_INT0     ; Переход на  обработку запроса IRQ0
$0004   jmp EXT_INT1     ; Переход на  обработку запроса IRQ1
$0006   jmp EXT_INT2     ; Переход на  обработку запроса IRQ2
$0008   jmp EXT_INT3     ; Переход на  обработку запроса IRQ3
$000A   jmp EXT_INT4     ; Переход на  обработку запроса IRQ4
```

```
$000C   jmp EXT_INT5     ; Переход на  обработку запроса IRQ5
$000E   jmp EXT_INT6     ; Переход на  обработку запроса IRQ6
$0010   jmp EXT_INT7     ; Переход на  обработку запроса IRQ7
$0012   jmp TIM2_COMP    ; Переход на обработку при выполнении условия
сравнения таймера 2
$0014   jmp TIM2_OVF     ; Переход на обработку при переполнении  таймера 2
$0016   jmp TIM1_CAPT    ; Переход на обработку при захвате фронта таймером
1
$0018   jmp TIM1_COMPA   ; Переход на обработку при срабатывании компаратора
A таймера 1
$001A   jmp TIM1_COMPB   ; Переход на обработку при срабатывании компаратора
B таймера 1
$001C   jmp TIM1_OVF     ; Переход на обработку при переполнении таймера 1
$001E   jmp TIM0_COMP    ; Переход на обработку при выполнения условия
сравнения таймера 0
$0020   jmp TIM0_OVF     ;Переход на обработку при переполнении таймера 0
$0022   jmp SPI_STC      ; Переход на обработку при завершении передачи SPI
$0024   jmp USART0_RXC   ; Переход на обработку при завершении приема УСАПП0
$0026   jmp USART0_DRE   ; Переход на обработку при освобождении регистра
данных UDR УСАПП0
$0028   jmp USART0_TXC   ; Переход на обработку при завершении передачи
УСАПП0
$002A   jmp ADC          ; Переход на обработку при завершении преобразования
АЦП
$002C   jmp EE_RDY       ; Переход на обработку при готовности ЭСППЗУ
$002E   jmp ANA_COMP     ; Переход на обработку при срабатывании аналогового
компаратора
$0030   jmp TIM1_COMPC   ; Переход на обработку при срабатывании компаратора
C таймера 1
$0032   jmp TIM3_CAPT    ; Переход на обработку при захвате фронта таймером 3
$0034   jmp TIM3_COMPA   ; Переход на обработку при срабатывании компаратора
A таймера 3
$0036   jmp TIM3_COMPB   ; Переход на обработку при срабатывании компаратора
B таймера 3
$0038   jmp TIM3_COMPC   ; Переход на обработку при срабатывании компаратора
C таймера 3
$003A   jmp TIM3_OVF     ; Переход на обработку при переполнении таймера 3
$003C   jmp USART1_RXC   ; Переход на обработку по завершении приема УСАПП1
$003E   jmp USART1_DRE   ; Переход на обработку при освобождении регистра
данных UDR УСАПП1
$0040   jmp USART1_TXC   ; Переход на обработку при завершении передачи
УСАПП1
$0042   jmp TWI          ; Переход на обработку прерывания по двухпроводному
последовательному интерфейсу
$0044   jmp SPM_RDY      ; Переход на обработку прерывания при готовности
выполнения команды SPM
;
$0046   RESET:ldir16, high(RAMEND); Начало основной программы
$0047   out SPH,r16      ; Установка указателя стека в конце ОЗУ
$0048   ldi r16, low(RAMEND)
$0049   out SPL,r16
$004A   sei ; Разрешение прерываний
$004B  xxx
... ... ... ...
```

Если конфигурационный бит BOOTRST незапрограммирован, размер загрузочного сектора установлен 8 кбайт и бит IVSEL установлен в регистре MCUCR 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 конце ОЗУ
$0002   ldi r16,low(RAMEND)
$0003   out SPL,r16
$0004   sei ; Разрешение прерываний
$0005  xxx
;
.org $F002
$F002   jmp EXT_INT0    ; Переход на обработку прерывания IRQ0
$F004   jmp EXT_INT1    ; Переход на обработку прерывания IRQ1
... ... ... ;
$F044   jmp SPM_RDY     ; Переход на обработку прерывания по готовности к
записи в память программ
```

Если конфигурационный бит BOOTRST запрограммирован и установлен размер загрузочного сектора 8 кбайт, то можно использовать следующий шаблон программы:

```
Адрес   Инструкция     Комментарий
.org $0002
$0002   jmp EXT_INT0   ; Переход на обработку прерывания IRQ0
$0004   jmp EXT_INT1   ; Переход на обработку прерывания IRQ1
... ... ... ;
$0044   jmp SPM_RDY    ; Переход на обработку прерывания по готовности к
записи в память программ
;
.org $F000
$F000   RESET: ldi r16,high(RAMEND) ; Начало основной программы
$F001   out SPH,r16    ; Установка указателя стека в конец ОЗУ
$F002   ldi r16,low(RAMEND)
$F003   out SPL,r16
$F004   sei            ; Разрешение прерываний
$F005  xxx
```

Если конфигурационный бит BOOTRST запрограммирован, размер загрузочного сектора установлен 8 кбайт и бит IVSEL в регистре MCUCR установлен перед разрешение любого из прерываний, то распределение адресов в программе следующее:

```
Адрес   Инструкция     Комментарий
;
.org $F000
$F000   jmp RESET      ; Переход на обработку сброса
$F002   jmp EXT_INT0   ; Переход на обработку прерывания IRQ0
$F004   jmp EXT_INT1   ; Переход на обработку прерывания IRQ1
... ... ... ;
$F044   jmp SPM_RDY    ; Переход на обработку прерывания по готовности записи
в память программ
$F046   RESET: ldi r16,high(RAMEND) ; Начало основной программы
$F047   out SPH,r16    ; Установка указателя стека в конец ОЗУ
$F048   ldi r16,low(RAMEND)
$F049   out SPL,r16
$F04A   sei            ; Разрешение прерываний
$F04B  xxx
```

Перемещение между секторами загрузочной и прикладной программы

Общий регистр управления прерываниями задает размещение таблицы векторов прерываний.

Регистр управления микроконтроллером – MCUCR

Разряд	7	6	5	4	3	2	1	0	
	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	MCUCR
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 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.

Разряд 0 – 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 MCUCR, r16
; Перемещение векторов в загрузочный сектор флэш-памяти
ldi r16, (1<<IVSEL)
out MCUCR, r16
ret
```

Пример кода на Си

```
void Move_interrupts(void)
{
/* Разрешение изменения векторов прерываний */
MCUCR = (1<<IVCE);
/* Перемещение векторов в загрузочной сектор флэш-памяти */
MCUCR = (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 29. Подробный перечень параметров ПВВ приведен в разделе "Электрические характеристики".

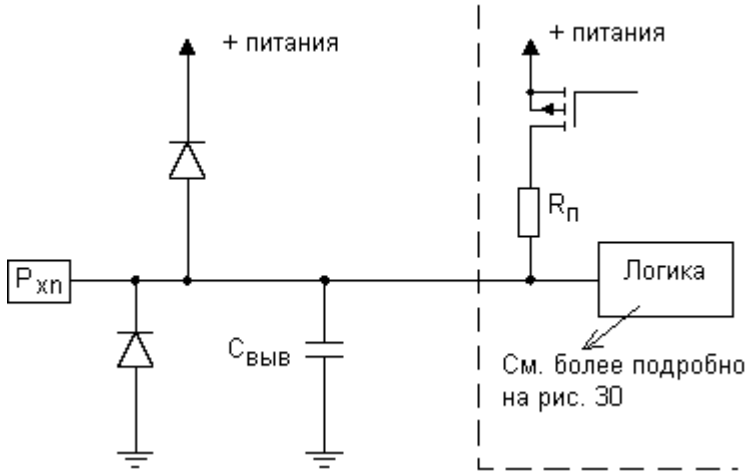


Рисунок 29 – Эквивалентная схема линии ПВВ

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).

Обратите внимание, что для некоторых портов разрешение альтернативных функций некоторых выводов делает невозможным использование других выводов для универсального цифрового ввода-вывода.

Порты в качестве универсального цифрового ввода-вывода

Все порты являются двунаправленными портами ввода-вывода с опциональными подтягивающими резисторами. Рисунок 30 иллюстрирует функциональную схему одной линии порта ввода-вывода, обозначенный как Pxn.

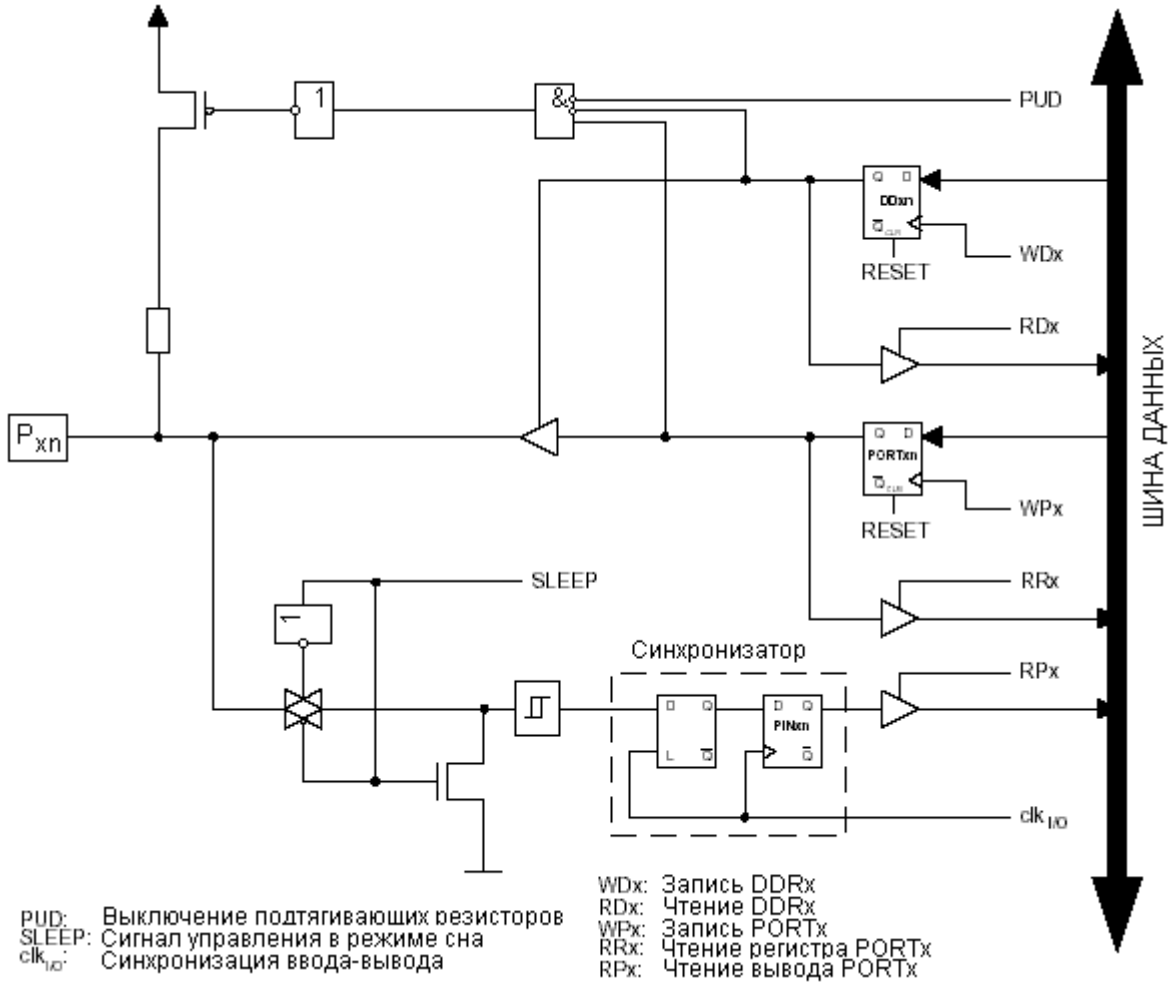


Рисунок 30 – Организация универсального цифрового ввода-вывода (1)

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.

Если PORTxn = 1 при конфигурации линии порта на вывод, то 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 25 подытоживается действие управляющих сигналов на состояние вывода.

Таблица 25 – 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 30 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 31 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.

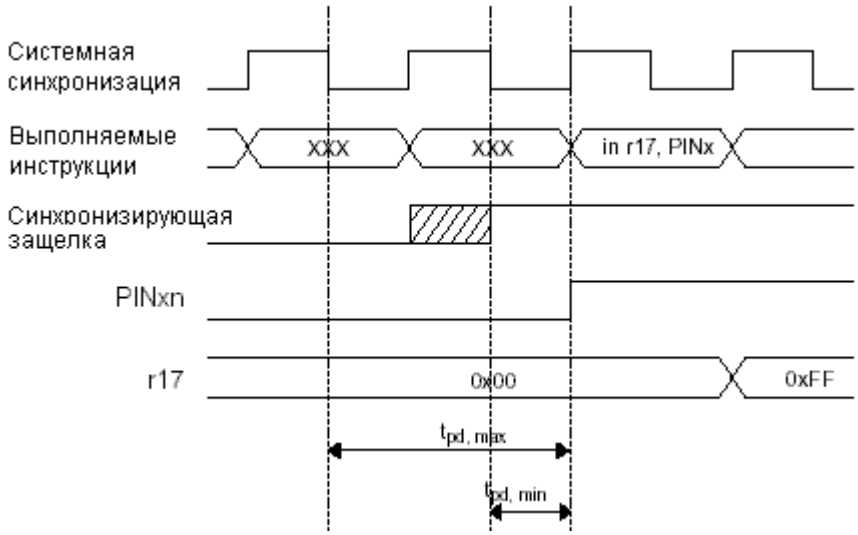


Рисунок 31 – Синхронизация во время опроса приложенного к выводу порта уровня

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 30 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 33 показано как управляющие сигналы, представленные на упрощенном рисунке 30, 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.

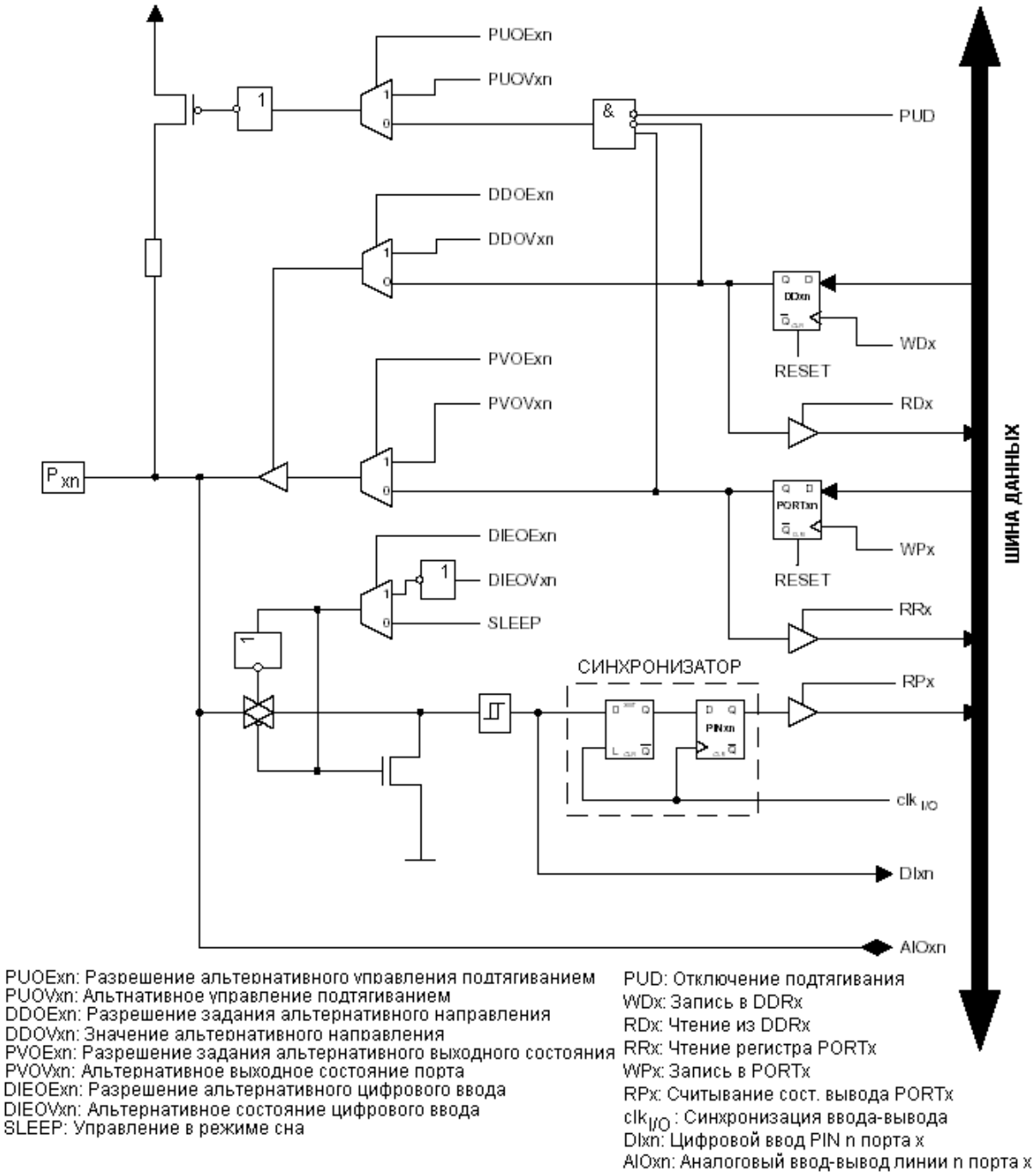


Рисунок 33 – Альтернативные функции порта (1)

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 26 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 26 – 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	

Разряд 2 – 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 A

Альтернативной функцией порта A является мультиплексированная младшая шина адреса/шина данных внешнего интерфейса памяти.

Таблица 27 – Альтернативные функции выводов порта A

Вывод порта	Альтернативная функция
PA7	AD7 (Разряд 7 шины адреса и шины данных внешнего интерфейса памяти)
PA6	AD6 (Разряд 6 шины адреса и шины данных внешнего интерфейса памяти)
PA5	AD5 (Разряд 5 шины адреса и шины данных внешнего интерфейса памяти)
PA4	AD4 (Разряд 4 шины адреса и шины данных внешнего интерфейса памяти)
PA3	AD3 (Разряд 3 шины адреса и шины данных внешнего интерфейса памяти)
PA2	AD2 (Разряд 2 шины адреса и шины данных внешнего интерфейса памяти)
PA1	AD1 (Разряд 1 шины адреса и шины данных внешнего интерфейса памяти)
PA0	AD0 (Разряд 0 шины адреса и шины данных внешнего интерфейса памяти)

В таблицах 28 и 29 приведена связь отключающих сигналов, представленных на рис. 33, и альтернативных функций выводов порта A.

Таблица 28- Отключающие сигналы для разрешения альтернативных функций на PA7..PA4

Наименование сигнала	PA7/AD7	PA6/AD6	PA5/AD5	PA4/AD4
PUOE	SRE	SRE	SRE	SRE
PUOV	~(WR | ADA (1) • PORTA7 • PUD	~(WR | ADA) • PORTA6 • PUD	~(WR | ADA) • PORTA5 • PUD	~(WR | ADA) • PORTA4 • PUD
DDOE	SRE	SRE	SRE	SRE
DDOV	WR | ADA	WR | ADA	WR | ADA	WR | ADA
PVOE	SRE	SRE	SRE	SRE
PVOV	A7 • ADA | D7 OUTPUT • WR	A6 • ADA | D6 OUTPUT • WR	A5 • ADA | D5 OUTPUT • WR	A4 • ADA | D4 OUTPUT • WR
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	D7 INPUT	D6 INPUT	D5 INPUT	D4 INPUT
AIO	-	-	-	-

Прим. 1: ADA существует короткий интервал времени, когда выводятся адресные сигналы (см. также “Внешний интерфейс памяти”).

Таблица 29- Отключающие сигналы для разрешения альтернативных функций на PA3..PA0

Наименование сигнала	PA3/AD3	PA2/AD2	PA1/AD1	PA0/AD0
PUOE	SRE	SRE	SRE	SRE
PUOV	~(WR | ADA) • PORTA3 • PUD	~(WR | ADA) • PORTA2 • PUD	~(WR | ADA) • PORTA1 • PUD	~(WR | ADA) • PORTA0 • PUD
DDOE	SRE	SRE	SRE	SRE
DDOV	WR | ADA	WR | ADA	WR | ADA	WR | ADA
PVOE	SRE	SRE	SRE	SRE
PVOV	A3 • ADA | D3 OUTPUT • WR	A2 • ADA | D2 OUTPUT • WR	A1 • ADA | D1 OUTPUT • WR	A0 • ADA | D0 OUTPUT • WR
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	D3 INPUT	D2 INPUT	D1 INPUT	D0 INPUT
AIO	-	-	-	-

Альтернативные функции порта B

Выводы порта B с альтернативными функциями показаны в таблице 30.

Таблица 30 – Альтернативные функции порта B

Вывод порта	Альтернативная функция
PB7	OC2/OC1C(1) (выход компаратора и выход ШИМ таймера-счетчика 2 или выход C компаратора и ШИМ таймера-счетчика 1)
PB6	OC1B (выход B компаратора и ШИМ таймера-счетчика 1)
PB5	OC1A (выход A компаратора и ШИМ таймера-счетчика 1)
PB4	OC0 (Выход компаратора и ШИМ таймера-счетчика 0)
PB3	MISO (Ввод для ведущей/вывод для подчиненной шины SPI)
PB2	MOSI (Вывод для ведущей/ввод для подчиненной шины SPI)
PB1	SCK (Синхронизация последовательной связи шины SPI)
PB0	SS (вход выбора подчиненного режима интерфейса SPI)

Прим. 1: OC1C отсутствует в режиме совместимости с ATmega103.

Ниже дано описание альтернативных функций выводов:

OC2/OC1C, разряд 7

OC2 – выход компаратора таймера-счетчика 2. Для выполнения данной функции вывод PB7 конфигурируется как выход (DDB7 = 1). Вывод OC2 также выполняет функцию выхода, когда таймер переводится в режим ШИМ.

OC1C – выход компаратора C таймера-счетчика 1. Для выполнения данной функции вывод PB7 настраивается как выход (DDB7 = 1). Вывод OC1C также выполняет функцию выхода, когда таймер переведен в режим ШИМ.

OC1B, разряд 6

OC1B – выход компаратора B таймера-счетчика 1. Для выполнения данной функции вывод PB6 настраивается как выход (DDB6 = 1). Вывод OC1B также выполняет функцию выхода, когда таймер переведен в режим ШИМ.

OC1A, разряд 5

OC1A – выход компаратора A таймера-счетчика 1. Для выполнения данной функции вывод PB5 настраивается как выход (DDB5 = 1). Вывод OC1A также выполняет функцию выхода, когда таймер переведен в режим ШИМ.

OC0, разряд 4

OC0 – выход компаратора таймера-счетчика 0. Для выполнения данной функции вывод PB4 настраивается как выход (DDB4 = 1). Вывод OC0 также выполняет функцию выхода, когда таймер переведен в режим ШИМ.

MISO – порт B, разряд 3

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 DDB3. Если работа SPI разрешена как подчиненного, то направление передачи данных задается DDB3. Если вывод принудительно настраивается на ввод, то подключение подтягивающего резистора останется под управлением бита PORTB3.

MOSI – порт B, разряд 2

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 DDB2. Если работа SPI разрешена как ведущего (мастера), направление передачи данных определяется DDB2. Если вывод принудительно настраивается как вход, то подключение подтягивающего резистора останется под управлением PORTB2.

SCK – порт B, разряд 1

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 DDB1. Если работа SPI разрешена как ведущего, то направление передачи данных управляется DDB1. Если вывод принудительно настроен на ввод, то управление подтягивающими резисторами осуществляется битом PORTB1.

SS – порт B, разряд 0

SS - вход выбора подчиненного порта. Если работа SPI разрешена как подчиненного, то данный вывод настраивается на ввод независимо от установки DDB0. 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 DDB0. Если вывод принудительно настроить как вход, то подключение подтягивающего резистора управляется битом PORTB0.

В таблицах 31 и 32 показаны значения отключающих сигналов (см. рис. 33) 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 31 – Отключающие сигналы для альтернативных функций на PB7..PB4

Наименование сигнала	PB7/OC2/OC1C	PB6/OC1B	PB5/OC1A	PB4/OC0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	Разрешение OC2/OC1C (1)	Разрешение OC1B	Разрешение OC1A	Разрешение OC0
PVOV	OC2/OC1C(1)	OC1B	OC1A	OC0B
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	-	-	-	-

Прим. 1: См. “Модулятор выхода компаратора (OCM1C2)” для изучения деталей. OC1C не существует в режиме совместимости с ATmega103.

Таблица 32 – Отключающие сигналы для альтернативных функций на PB3..PB0

Наименование сигнала	PB3/MISO	PB2/MOSI	PB1/SCK	PB0/SS
PUOE	SPE•MSTR	SPE•MSTR	SPE•MSTR	SPE•MSTR
PUOV	PORTB3•PUD	PORTB2•PUD	PORTB1•PUD	PORTB0•PUD
DDOE	SPE • MSTR	SPE•MSTR	SPE•MSTR	SPE•MSTR
DDOV	0	0	0	0
PVOE	SPE•MSTR	SPE•MSTR	SPE•MSTR	0
PVOV	SPI SLAVE OUTPUT	SPI MSTR OUTPUT	SCK OUTPUT	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	-	-	-	-

Альтернативные функции порта C

В режиме совместимости с ATmega103 порт C работает только на вывод. Альтернативной функцией порта C является старшая шина адреса внешнего интерфейса памяти.

Таблица 33 – Альтернативные функции выводов порта C

Вывод порта	Альтернативная функция
PC7	A15 (Разряд 15 шины адреса внешнего интерфейса памяти)
PC6	A14 (Разряд 14 шины адреса внешнего интерфейса памяти)
PC5	A13 (Разряд 13 шины адреса внешнего интерфейса памяти)
PC4	A12 (Разряд 12 шины адреса внешнего интерфейса памяти)
PC3	A11 (Разряд 11 шины адреса внешнего интерфейса памяти)
PC2	A10 (Разряд 10 шины адреса внешнего интерфейса памяти)
PC1	A9 (Разряд 9 шины адреса внешнего интерфейса памяти)

PC0	A8 (Разряд 8 шины адреса внешнего интерфейса памяти)

Таблицы 34 и 35 показывают связь между альтернативными функциями выводов порта и отключающими сигналами, показанных на рисунке 33.

Таблица 34 – Отключающие сигналы для разрешения альтернативных функций на PC7..PC4

Наименование сигнала	PC7/A15	PC6/A14	PC5/A13	PC4/A12
PUOE	SRE • (XMM(1)<1)	SRE • (XMM<2)	SRE • (XMM<3)	SRE • (XMM<4)
PUOV	0	0	0	0
DDOE	SRE • (XMM<1)	SRE • (XMM<2)	SRE • (XMM<3)	SRE • (XMM<4)
DDOV	1	1	1	1
PVOE	SRE • (XMM<1)	SRE • (XMM<2)	SRE • (XMM<3)	SRE • (XMM<4)
PVOV	A11	A10	A9	A8
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	-	-	-	-

Прим. 1: XMM = 0 в режиме совместимости с ATmega103.

Таблица 35 – Отключающие сигналы для разрешения альтернативных функций на PC3..PC0 (1)

Наименование сигнала	PC3/A11	PC2/A10	PC1/A9	PC0/A8
PUOE	SRE • (XMM<5)	SRE • (XMM<6)	SRE • (XMM<7)	SRE • (XMM<7)
PUOV	0	0	0	0
DDOE	SRE • (XMM<5)	SRE • (XMM<6)	SRE • (XMM<7)	SRE • (XMM<7)
DDOV	1	1	1	1
PVOE	SRE • (XMM<5)	SRE • (XMM<6)	SRE • (XMM<7)	SRE • (XMM<7)
PVOV	A11	A10	A9	A8
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	-	-	-	-

Прим. 1: XMM = 0 в режиме совместимости с ATmega103.

Альтернативные функции порта D

Выводы порта D с альтернативными функциями представлены в таблице 36.

Таблица 36 – Альтернативные функции выводов порта D

Вывод порта	Альтернативная функция
PD7	T2 (вход синхронизации таймера-счетчика 2)
PD6	T1 (вход синхронизации таймера-счетчика 1)

PD5	XCK1(1) (вход/выход внешней синхронизации УСАПП1)
PD4	IC1 (вход триггера захвата фронта таймера-счетчика 1)
PD3	INT3/TXD1(1) (вход внешнего прерывания 3 или выход передачи УАПП1)
PD2	INT2/RXD1(1) (вход внешнего прерывания 2 или вход приема УАПП1)
PD1	INT1/SDA(1) (вход внешнего прерывания 1 или ввод/вывод последовательных данных TWI)
PD0	INT0/SCL(1) (вход внешнего прерывания 0 или синхронизация последовательной связи TWI)

Прим. 1: XCK1, TXD1, RXD1, SDA и SCL отсутствуют в режиме совместимости с ATmega103.

Ниже приведены альтернативные конфигурации выводов порта:

T2 – порт D, разряд 7

T2 – счетный вход таймера-счетчика 2.

T1 – порт D, разряд 6

T1 – счетный вход таймера-счетчика 1.

XCK1 – порт D, разряд 4

XCK1 – внешняя синхронизация УСАПП1. Регистр направления данных (DDD4) задает является ли синхронизация выходной (DDD4=1) или входной (DDD4=0). Вывод XCK1 активен только если УСАПП1 работает в синхронном режиме.

IC1 – порт D, разряд 4

IC1 – вход захвата фронта таймера-счетчика 1.

INT3/TXD1 – порт D, разряд 3

INT3 – источник внешнего прерывания 3. Вывод PD3 может использоваться как источник внешнего прерывания микроконтроллера.

TXD1 – передача данных (вывод данных для УСАПП1). Если работа передатчика УСАПП1 разрешена, то данный вывод настраивается как выход независимо от значения DDD3.

INT2/RXD1 – порт D, разряд 2

INT2 – источник внешнего прерывания 2. Вывод PD2 может использоваться как источник внешнего прерывания микроконтроллера.

RXD1 – прием данных (ввод данных для УСАПП1). Если работа приемника УСАПП1 разрешена, то данный вывод настраивается на ввод независимо от значения DDD2. После перевода УСАППом данного вывода на вход, управление подтягивающим резистором осуществляется битом PORTD2.

INT1/SDA – порт D, разряд 1

INT1 – источник внешнего прерывания 1. Вывод PD1 может использоваться как источник внешнего прерывания микроконтроллера.

SDA – ввод-вывод данных двухпроводного последовательного интерфейса TWI. После установки бита TWEN в регистре TWCR разрешается работа двухпроводного последовательного

интерфейса, вывод PD1 отключается от порта и становится линией ввода-вывода последовательных данных 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.

INT0/SCL – порт D, разряд 0

INT0 – источник внешнего прерывания 0. Вывод PD0 может использоваться как источник внешнего прерывания микроконтроллера.

SCL – синхронизация двухпроводного последовательного интерфейса. Если установлен бит TWEN в регистре TWCR, то разрешается работа двухпроводного последовательного интерфейса, вывод PD0 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.

Таблицы 37 и 38 показывают связь между альтернативными функциями выводов порта D и отключающими сигналами, показанными на рисунке 33.

Таблица 37 – Отключающие сигналы для разрешения альтернативных функций на PD7..PD4

Наименование сигнала	PD7/T2	PD6/T1	PD5/XCK1	PD4/IC1
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	UMSEL1	0
PVOV	0	0	XCK1 OUTPUT	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	T2 INPUT	T1 INPUT	XCK1 INPUT	IC1 INPUT
AIO	-	-	-	-

Таблица 38 – Отключающие сигналы для разрешения альтернативных функций на PD3..PD0

Наименование сигнала	PD3/INT3/TXD1	PD2/INT2/RXD1	PD1/INT1/SDA	PD0/INT0/SCL
PUOE	TXEN1	RXEN1	TWEN	TWEN
PUOV	0	PORTD2•PUD	PORTD1•PUD	PORTB0•PUD
DDOE	SPE • MSTR	SPE•MSTR	SPE•MSTR	SPE•MSTR
DDOV	TXEN1	RXEN1	TWEN	TWEN
PVOE	1	0	SDA_OUT	SCL_OUT
PVOV	TXEN1	0	TWEN	TWEN
DIEOE	INT3 ENABLE	INT2 ENABLE	INT1 ENABLE	INT0 ENABLE
DIEOV	1	1	1	1
DI	INT3 INPUT	INT2 INPUT/RXD1	INT1 INPUT	INT0 INPUT
AIO	-	-	SDA INPUT	SCL INPUT

Прим. 1: После разрешения работы TWI активизируется схема управления скоростью изменения выходных сигналов на выводах PD0 и PD1. Данная функция не учтена в таблице. Кроме того, помехоподавляющие фильтры подключены между выходами AIO (см. рисунок 33) и цифровой логикой модуля TWI.

Альтернативные функции порта E

Альтернативные функции порта E представлены в таблице 39.

Таблица 39 – Альтернативные функции выводов порта E

Вывод порта	Альтернативная функция
PE7	INT7/IC3(1) (вход внешнего прерывания 7 или вход триггера захвата фронта таймера-счетчика 3)
PE6	INT6/ T3(1) (вход внешнего прерывания 6 или вход синхронизации таймера-счетчика 3)
PE5	INT5/OC3C(1) (вход внешнего прерывания 5 или выход C компаратора и ШИМ таймера-счетчика 3)
PE4	INT4/OC3B(1) (вход внешнего прерывания 4 или выход B компаратора и ШИМ таймера-счетчика 3)
PE3	AIN1/OC3A (1) (инвертирующий вход аналогового компаратора или выход A компаратора и ШИМ таймера-счетчика 3)
PE2	AIN0/XCK0(1) (неинвертирующий вход аналогового компаратора или вход/выход внешний синхронизации УСАПП0)
PE1	PDO/TXD0 (вывод программируемых данных или вывод передачи УАПП0)
PE0	PDI/RXD0 (ввод программируемых данных или вывод приема УАПП0)

Прим.: 1. IC3, T3, OC3C, OC3B, OC3B, OC3A и XCK0 отсутствуют в режиме совместимости с ATmega103.

INT7/IC3 – Порт E, разряд 7

INT7 – Источник внешнего прерывания 7. Вывод PE7 может выполнять функцию источника внешнего прерывания микроконтроллера.

IC3 – вход захвата фронтов таймера-счетчика 3.

INT6/T3 – Порт E, разряд 6

INT6 – Источник внешнего прерывания 6. Вывод PE6 может выполнять функцию источника внешнего прерывания микроконтроллера.

T3 – Счетный вход таймера-счетчика 3.

INT5/OC3C – Порт E, разряд 5

INT5 – Источник внешнего прерывания 5. Вывод PE5 может выполнять функцию источника внешнего прерывания микроконтроллера.

OC3C – выход компаратора C таймера-счетчика 3. Для выполнения данной функции вывод должен быть настроен как выход (DDE5 =1). Вывод OC3C также выполняет функцию выхода, когда таймер переведен в режим ШИМ.

INT4/OC3B – Порт E, разряд 4

INT4 – Источник внешнего прерывания 4. Вывод PE4 может выполнять функцию источника внешнего прерывания микроконтроллера.

OC3B – выход компаратора B таймера-счетчика 3. Для выполнения данной функции вывод должен быть настроен как выход (DDE4 =1). Вывод OC3B также выполняет функцию выхода, когда таймер переведен в режим ШИМ.

AIN1/OC3A – Порт E, разряд 3

AIN1 – инвертирующий вход аналогового компаратора. Данный вывод непосредственно подключен к инвертирующему входу аналогового компаратора.

OC3A – выход компаратора A таймера-счетчика 3. Для выполнения данной функции вывод должен быть настроен как выход (DDE3 =1). Вывод OC3A также выполняет функцию выхода, когда таймер переведен в режим ШИМ.

AIN0/XCK0 – Порт E, разряд 2

AIN0 – неинвертирующий вход аналогового компаратора. Данный вывод непосредственно подключен к неинвертирующему входу аналогового компаратора.

XCK0, USART0 – внешняя синхронизация. Регистр направления данных (DDE2) задает, является ли синхронизация выходной (DDE2=1) или входной (DDE2=0). Вывод XCK0 активен только тогда, когда УСАПП0 работает в синхронном режиме.

PDO/TXD0 – Порт E, разряд 1

PDO – вывод последовательно программируемых через SPI данных. В процессе последовательного программирования данный вывод используется как линия вывода данных из ATmega128.

TXD0 – вывод передачи УАПП0.

PDI/RXD0 – Порт E, разряд 0

PDI – ввод последовательно программируемых через SPI данных. В процессе последовательного программирования данный вывод используется как линия ввода данных в ATmega128.

RXD0 – Вывод приема данных УСАПП0. Если разрешена работа приемника УСАПП0, то данный вывод настраивается как вход независимо от состояния DDRE0. После того, как УСАПП0 настроит данный вывод как вход, запись лог. 1 в PORTE0 включит подтягивающий резистор на данном выводе.

В таблицах 40 и 41 описывается связь альтернативных функций выводов порта E и отключающих сигналов, представленных на рисунке 33.

Таблица 40 – Отключающие сигналы для разрешения альтернативных функций на PE7..PE4

Наименование сигнала	PE7/INT7/IC3	PE6/INT6/T3	PE5/INT5/OC3C	PE4/INT4/OC3B
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0

PVOE	0	0	OC3C ENABLE	OC3B ENABLE
PVOV	0	0	OC3C	OC3B
DIEOE	INT7 ENABLE	INT6 ENABLE	INT5 ENABLE	INT4 ENABLE
DIEOV	1	1	1	1
DI	INT7 INPUT/IC3 INPUT	INT7 INPUT/T3 INPUT	INT5 INPUT	INT4 INPUT
AIO	-	-	-	-

Таблица 41 – Отключающие сигналы для разрешения альтернативных функций на PE3..PE0

Наименование сигнала	PE3/AIN1/OC3A	PE2/AIN0/XCK0	PE1/PDO/TXD0	PE0/PDI/RXD0
PUOE	0	0	TXEN0	RXEN0
PUOV	0	0	0	PORTD1•PUD
DDOE	0	0	TXEN0	RXEN0
DDOV	0	0	1	0
PVOE	OC3B ENABLE	UMSEL0	TXEN0	0
PVOV	OC3C	XCK0 OUTPUT	TXD0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	0	RXD0	-	RXD0
AIO	AIN1 INPUT	AIN0 INPUT	-	-

Альтернативные функции порта F

Альтернативной функцией порта F является аналоговый ввод к АЦП (см. табл. 42).

Если некоторые выводы порта F используются как выходы, то необходимо следить, чтобы во время преобразования АЦП не происходило их переключение. Иначе, результат преобразования может быть некорректным. В режиме совместимости с ATmega103 порт F работает только на ввод. Если разрешена работа интерфейса JTAG, то подтягивающие резисторы на выводах PF7(TDI), PF5(TMS) и PF4(TCK) остаются подключенными, даже если микроконтроллер переведен в состояние сброса.

Таблица 42 – Альтернативные функции выводов порта F

Вывод порта	Альтернативная функция
PF7	ADC7/TDI (Вход канала 7 АЦП или ввод данных при JTAG тестировании)
PF6	ADC6/TDO (Вход канала 6 АЦП или вывод данных при JTAG тестировании)
PF5	ADC5/TMS (Вход канала 5 АЦП или выбор режима JTAG тестирования)
PF4	ADC4/TCK (Вход канала 4 АЦП или синхронизация JTAG тестирования)
PF3	ADC3 (Вход канала 3 АЦП)
PF2	ADC2 (Вход канала 2 АЦП)
PF1	ADC1 (Вход канала 1 АЦП)
PF0	ADC0 (Вход канала 0 АЦП)

TDI, ADC7 – Порт F, разряд 7

ADC7 – Аналогово-цифровой преобразователь, канал 7.

TDI – Ввод данных при JTAG-тестировании. Последовательный ввод данных происходит в регистр инструкций или регистр данных (сканируемые звенья). После разрешения работы JTAG-интерфейса данный вывод не может использоваться в качестве линии ввода-вывода.

TDO, ADC6 – Порт F, разряд 6

ADC6 – Аналогово-цифровой преобразователь, канал 6.

TDO – вывод данных при JTAG-тестировании. Последовательный вывод данных из регистра инструкции или регистра данных. После разрешения работы JTAG-интерфейса данный вывод не может использоваться в качестве линии ввода-вывода. Вывод TDO становится тристабильным, если введено состояние TAP, при котором происходит сдвиг выводимых данных.

TMS, ADC5 – Порт F, разряд 5

ADC5 – Аналогово-цифровой преобразователь, канал 5.

TMS – Выбор режима JTAG тестирования. Данный вывод используется для управления цифровым автоматом TAP-контроллера. После разрешения работы JTAG-интерфейса данный вывод не может использоваться в качестве линии ввода-вывода.

TCK, ADC4 – Порт F, разряд 4

ADC4 – Аналогово-цифровой преобразователь, канал 4.

TCK – синхронизация JTAG-тестирования. Работа интерфейса JTAG синхронизирована с TCK. После разрешения работы JTAG-интерфейса данный вывод не может использоваться в качестве линии ввода-вывода.

ADC3 – ADC0 – Порт F, разряды 3..0

ADC7 – Аналогово-цифровой преобразователь, каналы 3…0.

Таблица 43 – Отключающие сигналы для разрешения альтернативных функций на PF7..PF4

Наименование сигнала	PF7/ADC7/TDI	PF6/ADC6/TDO	PF5/ADC5/TMS	PF4/ADC4/TCK
PUOE	JTAGEN	JTAGEN	JTAGEN	JTAGEN
PUOV	1	0	1	1
DDOE	JTAGEN	JTAGEN	JTAGEN	JTAGEN
DDOV	0	SHIFT_IR + SHIFT_DR	0	0
PVOE	0	JTAGEN	0	0
PVOV	0	TDO	0	0
DIEOE	JTAGEN	JTAGEN	JTAGEN	JTAGEN
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	TDI/ADC7 INPUT	ADC6 INPUT	TMS/ADC5 INPUT	TCK/ADC4 INPUT

Таблица 44 – Отключающие сигналы для разрешения альтернативных функций на PF3..PF0

Наименование сигнала	PF3/ADC3	PF2/ADC2	PF1/ADC1	PF0/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

Альтернативные функции порта G

В режиме совместимости с ATmega103 могут использоваться только альтернативные функции порта G, а функция универсального цифрового ввода-вывода не доступна. В таблице 45 приведены альтернативные функции порта G.

Таблица 45 – Альтернативные функции выводов порта G

Вывод порта	Альтернативная функция
PG4	TOSC1 (Генератор часов реального времени таймера-счетчика 0)
PG3	TOSC2 (Генератор часов реального времени таймера-счетчика 0)
PG2	(Разрешение фиксации адреса внешней памяти)
PG1	RD (Строб чтения внешней памяти)
PG0	WR (Строб записи внешней памяти)

TOSC1 – Порт G, разряд 4

TOSC1 – 1-ый вывод генератора таймера. После установки бита AS0 в регистре ASSR разрешается работа асинхронного тактирования таймера-счетчика 0, а вывод PG4 отключается от порта и становится входом инвертирующего усилителя генератора. В этом режиме кварцевый резонатор подключен к выводу PG4, который теперь не может использоваться как линия ввода-вывода.

TOSC2 – Порт G, разряд 3

TOSC2 – 2-ой вывод генератора таймера. После установки бита AS0 в регистре ASSR разрешается работа асинхронного тактирования таймера-счетчика 0, а вывод PG3 отключается от порта и становится инвертированным выходом усилителя генератора. В этом режиме кварцевый резонатор подключен к выводу PG3, который теперь не может использоваться как линия ввода-вывода.

ALE – Порт G, разряд 2

ALE – сигнал разрешения фиксации адреса внешней памяти.

RD – Порт G, разряд 1

RD – строб управления чтением данных внешней памяти.

WR – Порт G, разряд 0

WR – строб управления записью во внешнюю память.

В таблицах 46 и 47 представлена связь альтернативных функций порта G и отключающих сигналов, представленных на рисунке 33.

Таблица 46 – Отключающие сигналы для разрешения альтернативных функций на PG4..PG1

Наименование сигнала	PG4/TOSC1	PG3/TOSC2	PG2/ALE	PG1/RD
PUOE	AS0	AS0	SRE	SRE
PUOV	0	0	0	0
DDOE	AS0	AS0	SRE	SRE
DDOV	0	0	1	1
PVOE	0	0	SRE	SRE
PVOV	0	0	ALE	RD
DIEOE	AS0	AS0	0	0
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	T/C0 OSC INPUT	T/C0 OSC OUTPUT	-	-

Таблица 47 – Отключающий сигнал для разрешения альтернативной функции на PG0

Наименование сигнала	PG0/WR
PUOE	SRE
PUOV	0
DDOE	SRE
DDOV	1
PVOE	SRE
PVOV	WR
DIEOE	0
DIEOV	0
DI	–
AIO	–

Описание регистров портов ввода-вывода

Регистр данных порта A – PORTA

Разряд	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	**PORTA**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр направления данных порта A – DDRA

Разряд	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	**DDRA**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	

Исх. значение	0	0	0	0	0	0	0	0	

Адрес входов порта A – PINA

Разряд	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	**PINA**
Чтение/запись	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	
Исх. значение	-	-	-	-	-	-	-	-	

Регистр данных порта B – PORTB

Разряд	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	**PORTB**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр направления данных порта B – DDRB

Разряд	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	**DDRB**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Адрес входов порта B – PINB

Разряд	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	**PINB**
Чтение/запись	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	
Исх. значение	-	-	-	-	-	-	-	-	

Регистр данных порта C – PORTC

Разряд	7	6	5	4	3	2	1	0	
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	**PORTC**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр направления данных порта C – DDRC

Разряд	7	6	5	4	3	2	1	0	
	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	**DDRC**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Адрес входов порта C – PINC

Разряд	7	6	5	4	3	2	1	0	
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	**PINC**
Чтение/запись	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	
Исх. значение	-	-	-	-	-	-	-	-	

В режиме совместимости с ATmega103 регистры DDRC и PINC инициализируются для организации вывода лог. 0. Выводы порта принимают их исходное значение, даже если синхронизация не запущена. Обратите внимание, что регистры DDRC и PINC доступны в режиме совместимости с ATmega103 и не должны использоваться, если необходима 100%-ая совместимость снизу вверх.

Регистр данных порта D – PORTD

Разряд	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	**PORTD**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр направления данных порта D – DDRD

Разряд	7	6	5	4	3	2	1	0	
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	**DDRD**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Адрес входов порта D – PIND

Разряд	7	6	5	4	3	2	1	0	
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	**PIND**
Чтение/запись	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	
Исх. значение	-	-	-	-	-	-	-	-	

Регистр данных порта E – PORTE

Разряд	7	6	5	4	3	2	1	0	
	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0	**PORTE**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр направления данных порта E – DDRE

Разряд	7	6	5	4	3	2	1	0	
	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	**DDRE**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Адрес входов порта E – PINE

Разряд	7	6	5	4	3	2	1	0	
	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0	**PINE**
Чтение/запись	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	
Исх. значение	-	-	-	-	-	-	-	-	

Регистр данных порта F – PORTF

Разряд	7	6	5	4	3	2	1	0	
	PORTF7	PORTF6	PORTF5	PORTF4	PORTF3	PORTF2	PORTF1	PORTF0	**PORTF**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр направления данных порта F – DDRF

Разряд	7	6	5	4	3	2	1	0	
	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0	**DDRF**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Адрес входов порта F – PINF

Разряд	7	6	5	4	3	2	1	0	
	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0	**PINF**
Чтение/запись	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	
Исх. значение	-	-	-	-	-	-	-	-	

Обратите внимание, что регистры PORTF и DDRF не доступны в режиме совместимости с ATmega103, где порт F функционирует только как цифровой ввод.

Регистр данных порта G – PORTG

Разряд	7	6	5	4	3	2	1	0	
	-	-	-	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0	**PORTG**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр направления данных порта G – DDRG

Разряд	7	6	5	4	3	2	1	0	
	-	-	-	DDG4	DDG3	DDG2	DDG1	DDG0	**DDRG**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Адрес входов порта G – PING

Разряд	7	6	5	4	3	2	1	0	
	-	-	-	PING4	PING3	PING2	PING1	PING0	**PING**
Чтение/запись	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	
Исх. значение	-	-	-	-	-	-	-	-	

Обратите внимание, что PORTG, DDRG и PING не доступны в режиме совместимости с ATmega103. В режиме совместимости с ATmega103 порт G выполняет только свою альтернативную функцию (TOSC1, TOSC2, WR, RD и ALE).

Внешние прерывания

Внешние прерывания осуществляются через выводы INT7:0. Обратите внимание, что после разрешения внешние прерывания будут генерироваться, даже если линии INT7:0 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 регистрах управления внешними прерываниями EICRA (INT3:0) и EICRB (INT7:4). 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 INT7:4 необходимо наличие синхронизации ввода-вывода, описанной в разделе “Источники синхронизации и их распределение". Прерывания по низкому уровню и фронтам на INT3:0 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?С. 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.

Регистр А управления внешними прерываниями – EICRA

Разряд	7	6	5	4	3	2	1	0	
	ICS31	ICS30	ICS21	ICS20	ICS11	ICS10	ICS01	ICS00	**EICRA**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Данный регистр на доступен в режиме совместимости с ATmega103, но исходная установка соответствует настройке на прерывания по низкому уровню на INT3:0, что также как и в ATmega103.

Разряды 7..0 – ISC31, ISC30 – ISC00, ISC00: Биты выбора условия генерации внешнего прерывания 3 - 0

Внешние прерывания 3 - 0 активизируются через внешние выводы INT3:0, если установлены флаг I в регистре статуса SREG и соответствующая маска прерывания в EIMSK. Выбор уровня или фронта для активизации внешнего прерывания осуществляется в соответствии с таблицей 48. Фронты на INT3..INT0 выявляются асинхронно. Прерывание по выв. INT3:0 будет сгенерировано, если длительность импульса будет больше минимально необходимой (см. табл. 49). При возникновении импульсов меньшей длительности генерация прерывания не гарантируется. Если выбрано прерывание по низкому уровню, то для генерации прерывания необходимо, чтобы этот уровень оставался на прежнем низком уровне до момента завершения выполнения текущей инструкции. После разрешения прерывания по уровню оно будет генерироваться непрерывно до тех пор, пока на входе присутствует низкий уровень. При изменении бит ISCn может возникнуть прерывание. Поэтому, рекомендуется вначале отключить прерывание INTn путем сброса бита разрешения прерывания в регистре EIMSK. После этого, значение бит ISCn может быть изменено. И, наконец, перед возобновлением работы прерываний необходимо сбросить флаг прерывания INTn путем записи лог. 1 во флаг прерывания (INTFn) в регистре EIFR.

Таблица 48 – Задание условия генерации запроса на прерывание(1)

ISCn1	ISCn0	Описание
0	0	Низкий уровень на INTn генерирует запрос на прерывание
0	1	Зарезервировано
1	0	Падающий фронт на INTn генерирует асинхронно запрос на прерывание
1	1	Нарастающий фронт на INTn генерирует асинхронно запрос на прерывание

Прим.: 1. n = 3, 2, 1 или 0.

Перед изменением бит ISCn1/ISCn0 необходимо запретить работу прерывания путем очистки бита разрешения прерывания в регистре EIMSK. В противном случае прерывание может возникнуть после изменения данных бит.

Таблица 49 – Характеристики асинхронного внешнего прерывания

Обозначение	Параметр	Мин.	Типично	Макс.	Ед.изм.
tINT	Минимальная длительность импульса для генерации асинхронного прерывания		50		нс

Регистр B управления внешними прерываниями – EICRB

Разряд	7	6	5	4	3	2	1	0	
	ICS71	ICS70	ICS61	ICS60	ICS51	ICS50	ICS41	ICS40	**EICRB**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряды 7..0 – ISC71, ISC70 - ISC41, ISC40: Бита выбора условия генерации внешнего прерывания 7 - 4

Внешние прерывания 7 - 4 активизируются через внешние выводы INT7:4, если установлены флаг I в регистре статуса SREG и соответствующая маска прерывания в регистре EIMSK. Условие, по которому генерируется прерывание, выбирается исходя из данных таблицы 50. Для определения фронтов на выводах INT7:4 осуществляется выборка их состояний. Если выбрано прерывание по фронту или изменению уровня, то прерывание будет сгенерировано, если на входе появляется импульс, длительность которого больше одного периода синхронизации. При действии на входе более коротких импульсов генерация прерывания не гарантируется. Обратите внимание, что частота синхронизации ЦПУ может быть ниже чем частота XTAL, если разрешена работа делителя частоты XTAL. Если выбрано прерывание по низкому уровню, то прерывание генерируется, если до момента окончания выполнения текущей инструкции на входе по прежнему

присутствует низкий уровень. Если разрешено прерывание по уровню, то оно будет генерироваться непрерывно до тех пор, пока на входе присутствует низкий уровень.

Таблица 50 – Задание условия генерации запроса на прерывание(1)

ISCn1	ISCn0	Описание
0	0	Низкий уровень на INTn генерирует запрос на прерывание
0	1	Любое изменение логического состояния на INTn генерирует запрос на прерывание
1	0	Падающий фронт, выявленный по двум выборкам на INTn, генерирует запрос на прерывание.
1	1	Нарастающий фронт, выявленный по двум выборкам на INTn, генерирует запрос на прерывание.

Прим. 1: n = 7, 6, 5 или 4.

Перед изменением бит ISCn1/ISCn0 необходимо запретить работу прерывания путем очистки бита разрешения прерывания в регистре EIMSK. В противном случае прерывание может возникнуть после изменения данных бит.

Регистр маски внешнего прерывания – EIMSK

Разряд	7	6	5	4	3	2	1	0	
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	**EIMSK**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряды 7..0 – INT7 – INT0: Разрешение запроса на внешнее прерывание 7 - 0

Если в бит INT7 – INT0 и в бит I регистра статуса SREG записать лог. 1, то разрешается работа внешнего прерывания по соответствующему выводу. Биты выбора условия генерации прерывания в регистрах управления внешними прерываниями EICRA и EICRB определяют по какому условию генерируется прерывание: по нарастающему фронту, по падающему фронту или по уровню. Любой из данных выводов сохраняет активность, даже если он настроен на вывод. Данная особенность может использоваться для программной генерации прерывания.

Разряд	7	6	5	4	3	2	1	0	
	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0	**EIFR**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряды 7..0 – INTF7 - INTF0: Флаги внешних прерываний 7 - 0

Если фронт или изменение логического состояния на INT7:0 вызывает прерывание, то соответствующий флаг INTF7:0 принимает единичное состояние. Если установлены бит I регистра статуса SREG и соответствующий бит разрешения прерывания INT7:0 в регистре EIMSK, то микроконтроллер выполнит переход на вектор прерывания. Флаг сбрасывается аппаратно после выполнения процедуры обработки прерывания. Альтернативно флаг может быть сброшен программно путем записи лог. 1 в соответствующий бит. Если INT7:0 настроены на генерацию прерывания по уровню, то флаги постоянно находятся в сброшенном состоянии. Обратите внимание, что при переходе в режим сна с отключенными прерываниями INT3:0 входные буферы этих выводов будут отключенными. В свою очередь это может вызвать изменение внутреннего состояния сигналов, которое приведет к установке флагов INTF3:0. См. также “Разрешение цифрового ввода и режимы сна”.

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 107.

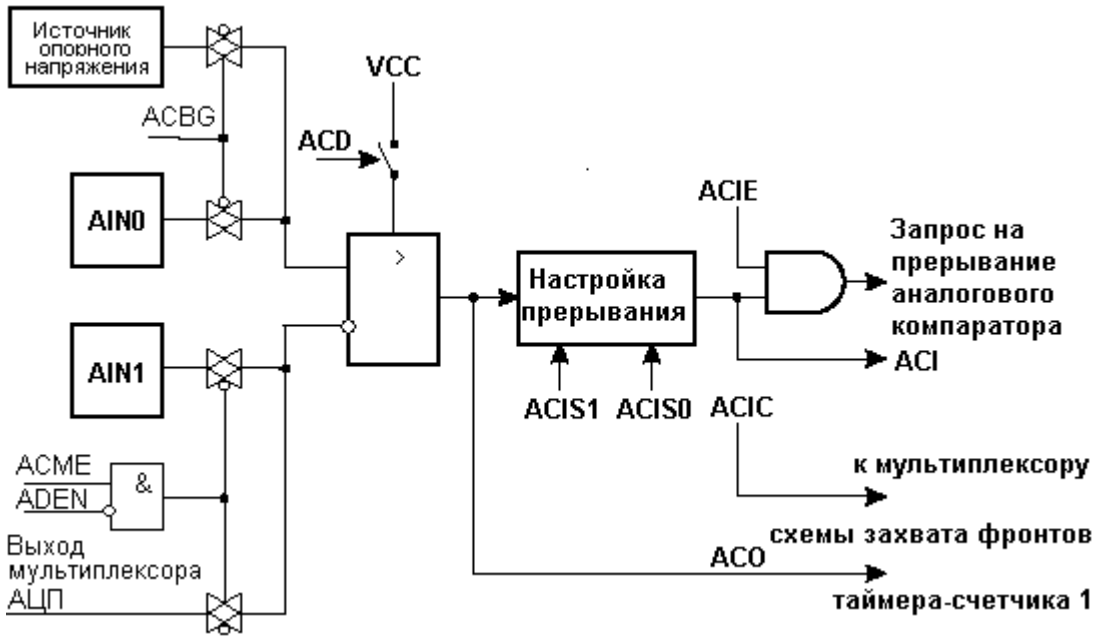


Рисунок 107 – Функциональная схема аналогового компаратора

Прим. 1: См. табл. 94.

Прим. 2: Информация по расположению выводов аналогового компаратора представлена на рисунке 1 и в таблице 30.

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	

Разряд 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”.

Регистр состояния и управления аналогового компаратора – ACSR

Разряд	7	6	5	4	3	2	1	0	
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Чтение/Запись	Чт./Зп.	Чт./Зп.	Чт.	Чт/Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Начальное знач.	0	0	x	0	0	0	0	0	

Разряд 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.

Разряд 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”.

Разряд 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.

Разряд 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.

Разряд 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.

Разряд 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).

Разряды 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. 93.

Таблица 93 - 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. 94). Если ACME сброшен или установлен ADEN, то в качестве неинвертирующего входа аналогового компаратора используется вывод микроконтроллера AIN1.

Таблица 94 – 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

16-разр. таймеры-счетчики 1 и 3

16-разрядные таймеры-счетчики предназначены для точного задания временных интервалов, генерации прямоугольных импульсов и измерения временных характеристик импульсных сигналов.

Основные отличительные особенности:

16-разрядные счетчики (в т.ч. возможность организации 16-разр. ШИМ)
Три 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 при записи нового порога сравнения в OCR (двойная буферизация) и фазовая коррекция

Переменный период ШИМ
Частотный генератор
Счетчик внешних событий
10 самостоятельных источников прерываний (TOV1, OCF1A, OCF1B, OCF1C, ICF1, TOV3, OCF3A, OCF3B, OCF3C и ICF3)

Ограничения на режим совместимости с ATmega103

Обратите внимание, что в режиме совместимости с ATmega103 доступен только один 16-разр. таймер-счетчик (таймер-счетчик 1). Кроме того, в режиме совместимости с ATmega103 таймер-счетчик 1 имеет только два регистра порогов сравнения (OCR1A и OCR1B).

Введение

В виду идентичности таймеров 1 и 3 в данном разделе используется общая форма записи. Так индекс “n” заменяет номер таймера-счетчика (1 или 3), а “x” заменяет наименование канала сравнения (A,B или C). 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 46. 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 таймеров представлены в параграфе “Описание регистров 16-разр. таймеров-счетчиков”.

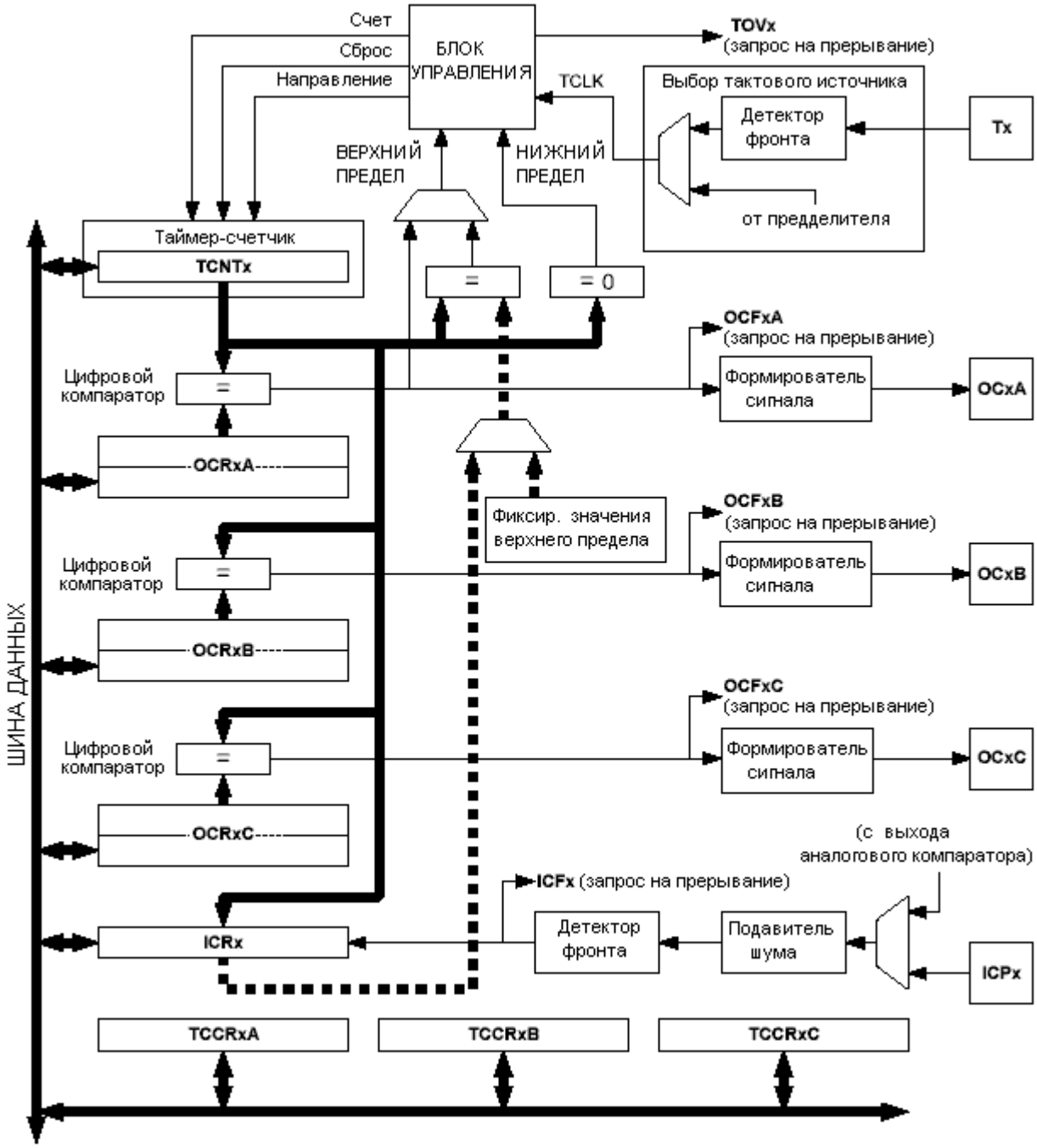


Рисунок 46- Функциональная схема 16-разр. таймера-счетчика

Прим.: Расположение и назначение выводов таймеров-счетчиков 1 и 3 см. на рисунке 1, таблице 30 и таблице 39.

Регистры

Регистр таймера-счетчика (TCNTn), регистры порогов сравнения (OCRnA/B/C), а также регистр захвата (ICRn) 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**). Регистры управления таймером (TCCRnA/B/C) 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) и регистром маски расширенных прерываний (ETIMSK). Регистры (E)TIFR и (E)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 Tn. 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 (clkTn).

Значение регистров порогов сравнения (OCRnA/B/C) непрерывно сравнивается со значением счетчика. Результат сравнения может использоваться для генерации прямоугольных импульсов с ШИМ или с переменной частотой на выходах OCnA/B/C. См. также “Блоки сравнения”. В случае определения совпадения значений сравниваемых регистров устанавливается соответствующий флаг прерываний (OCFnA/B/C), 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 ICPn или на выводах аналогового компаратора (см. “Аналоговый компаратор”). 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 OCRnA, ICRn или иметь фиксированные значения. Если OCRnA 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 ICRn, освобождая регистр OCRnA для функции широтно-импульсной модуляции.

Определения

Некоторые определения и их сокращенные наименования, которые интенсивно используются в этом разделе, представлены в таблице 57.

Таблица 57 – 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:

PWMn0 заменен на WGMn0.
PWMn1 заменен на WGMn1.
CTCn заменен на WGMn2.

Ниже приведены регистры, которые были добавлены к 16-разр. таймеру-счетчику:

Регистр управления таймером-счетчиком C (TCCRnC).
Регистр C порога сравнения, OCRnCH и OCRnCL (или OCRnC).

Следующие биты добавлены в регистры управления 16-разр. таймером-счетчиком:

COM1C1, COM1C0 добавлены в TCCR1A.
FOCnA, FOCnB и FOCnC добавлены в новый регистр TCCRnC.
WGMn3 добавлен в TCCRnB.
Добавлены флаг прерываний и биты маски прерываний для канала сравнения C.
Некоторые усовершенствования в ряде случаев затрагивают вопрос совместимости.

Доступ к 16-разр. регистрам

Регистры TCNTn, OCRnA/B/C и ICRn 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 OCRnA/B/C 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 OCRnA/B/C и ICRn.

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;
...
```

Прим: 1. При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр ввода-вывода расположен в расширенной памяти ввода-вывода, то инструкции “IN”, “OUT”, “SBIS”, “SBIC”, “CBI” и “SBI” необходимо заменить на инструкции доступа к расширенной памяти ввода-вывода “LDS” и “STS” совместно с инструкциями “SBRS”, “SBRC”, “SBR”, и “CBR”.

В примере на Ассемблере значение TCNTn 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 TCNTn без опасности изменения содержимого временного регистра в прерываниях. Аналогично данный пример следует распространять на доступ к регистрам OCRnA/B/C и ICRn.

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;
}
```

Прим: 1. При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр ввода-вывода расположен в расширенной памяти ввода-вывода, то инструкции “IN”, “OUT”, “SBIS”, “SBIC”, “CBI” и “SBI” необходимо заменить на инструкции доступа к расширенной памяти ввода-вывода “LDS” и “STS” совместно с инструкциями “SBRS”, “SBRC”, “SBR”, и “CBR”.

В коде на Ассемблере значение регистра TCNTn 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 TCNTn. На этом же принципе может быть выполнена запись в регистры OCRnA/B/C или ICRn.

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
```

```
    ; 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;
    }
```

Прим: 1. При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр ввода-вывода расположен в расширенной памяти ввода-вывода, то инструкции “IN”, “OUT”, “SBIS”, “SBIC”, “CBI” и “SBI” необходимо заменить на инструкции доступа к расширенной памяти ввода-вывода “LDS” и “STS” совместно с инструкциями “SBRS”, “SBRC”, “SBR”, и “CBR”.

В примере на Ассемблере предполагается, что записываемое значение в TCNTn 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 1/3

Таймер-счетчик может использовать как внешний, так и внутренний тактовые сигналы. Источник тактового сигнала выбирается соответствующей схемой микроконтроллера под управлением бит выбора синхронизации (CSn2:0), которые находятся в регистре B управления таймером-счетчиком (TCCRnB). Более подробная информация по тактовым источникам и предделителю приведена в разделе "Предделители таймера-счетчика 3, таймера-счетчика 2 и таймера-счетчика 1".

Блок счетчика

Основным элементом 16-разр. таймера-счетчика является программируемый реверсивный 16-разрядный счетчик. На рисунке 47 представлена функциональная схема счетчика и окружающих его элементов.

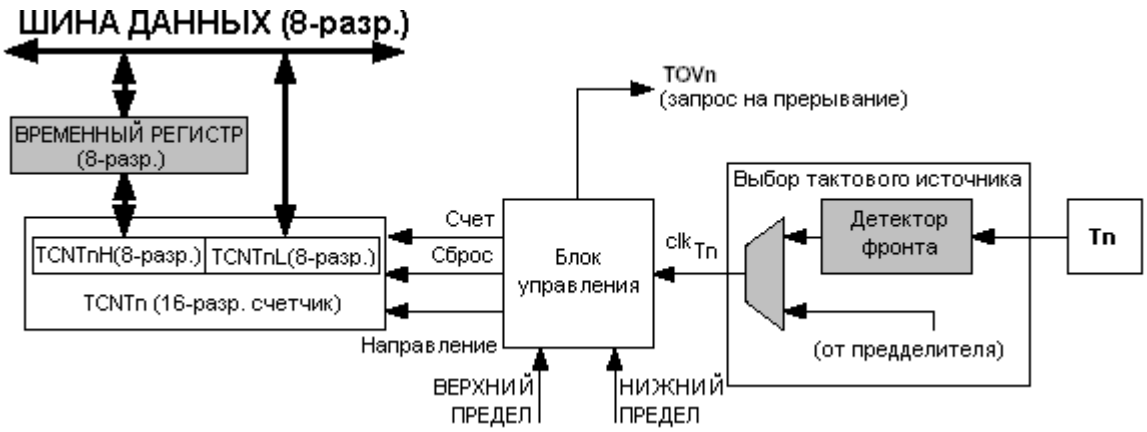


Рисунок 47 – Функциональная схема счетчика

Описание внутренних сигналов:

Счет – Инкрементирует или декрементирует состояние TCNTn на 1.

Направление – Задает прямой счет (инкрементирование) или обратный счет (декрементирование).

Сброс – Сброс TCNTn (установка всех разрядов к лог. 0).

clkTn – Синхронизация таймера-счетчика.

Верхний предел – Сигнализирует о достижении TCNTn максимального значения.

Нижний предел – Сигнализирует о достижении TCNTn минимального значения (нуля).

Содержимое 16-разр. счетчика разбито на две 8-разр. ячейки, расположенных в памяти ввода-вывода: Старший байт счетчика (TCNTnH) , в котором хранятся старшие 8-разрядов счетчика, и младший байт счетчика (TCNTnL), в котором хранятся младшие 8-разрядов. ЦПУ не имеет непосредственного доступа к регистру TCNTnH. Если ЦПУ выполняет доступ к TCNTnH, то фактически обращение происходит к временному регистру. Во временный регистр копируется значение TCNTnH, если выполняется чтение регистра TCNTnL и в TCNTnH копируется содержимое временного регистра, если выполняется запись в TCNTnL. 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 TCNTn во время счета счетчиком будет давать непредсказуемый результат. Такие случаи описаны в последующих параграфах.

В зависимости от используемого режима работы каждый такт синхронизации таймера clkTn счетчик будет сбрасываться, инкрементироваться или декрементироваться. Сигнал clkTn может быть внешним или внутренним, что задается битами выбора синхронизации (CSn2:0). Если тактовый источник не задан (CSn2:0 = 0), то таймер останавливается. Однако содержимое TCNTn остается доступным ЦПУ независимо от наличия синхронизации на clkTn. Если ЦПУ выполняет запись в TCNTn, то тем самым блокируется (запись имеет более высокий приоритет) любое действие счетчика: сброс или счет.

Алгоритм счета определяется значением бит режима работы таймера (WGMn3:0), расположенных в регистрах A и B управления таймером-счетчиком (TCCRnA и TCCRnB). Имеется четкая связь между алгоритмом счета счетчика и формой генерируемого на выходе OCnx сигнала. Боле подробная информация об этом приведена в “Режимы работы 16-разр. таймеров-счетчиков”.

Установка флага переполнения таймера-счетчика (TOVn) происходит в зависимости выбранного с помощью бит WGMn3:0 режима работы. Флаг TOVn 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 ICPn. 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 48. 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.

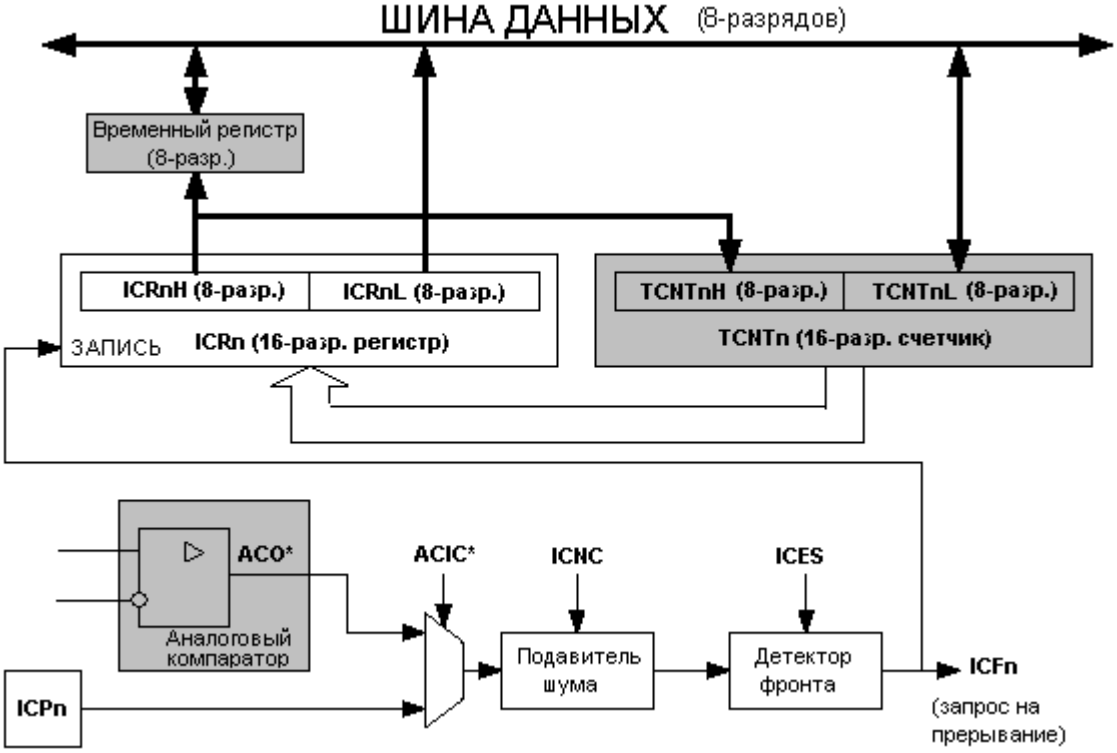


Рисунок 48 – Функциональная схема блока захвата

Прим.: Выход аналогового компаратора (ACO) только у таймера-счетчика 1 может выступать в качестве сигнала захвата. У таймера-счетчика 3 эта возможность отсутствует.

Если на входе захвата (ICPn) 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 (TCNTn) помещается в регистр захвата (ICRn). Флаг захвата (ICFn) устанавливается на том же такте ЦПУ, на котором произошло копирование значения TCNTn в ICRn. Установка бита TICIEn =1 разрешает прерывание по установке флага захвата. Флаг ICFn автоматически сбрасывается при переходе на вектор прерывания. Альтернативно флаг ICFn сбрасывается программно, если записать в него лог. 1. Считывание 16-разр. значения регистра захвата (ICRn) выполняется чтением сначала младшего байта (ICRnL), а затем старшего байта (ICRnH). При выполнении команды чтения младшего байта значение старшего байта автоматически копируется во временный регистр. Если ЦПУ выполняет команду чтения регистра ICRnH, то фактически считывается содержимое временного регистра. Запись в регистр ICRn возможна только в том случае, если битами задания режима работы таймера выбран режим, в котором значение регистра ICRn задает верхний предел счета. В этом случае необходимо выполнить соответствующую установку бит режима работы (WGMn3:0), а только затем выполнить запись значения верхнего предела в регистр ICRn. Запись 16-разр. значения в регистр ICRn выполняется путем записи сначала старшего байта в ICRnH, а только затем младшего байта в ICRnL (см. также “Доступ к 16-разр. регистрам”).

Источник срабатывания механизма захвата

Основным источником, инициирующим захват состояния таймер-счетчика, является вывод захвата (ICPn). 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 (ICPn) и выходом аналогового компаратора (ACO) выполнены по аналогии с каскадом, связанного с выв. Tn (см. рис. 59). 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 ICRn определяет вершину счета.

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

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 (ICNCn) в регистре B управления таймером-счетчиком (TCCRnB). 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 ICRn до момента возникновения следующего события, то регистр ICRn обновит свое содержимое. В этом случае результат захвата будет некорректным.

Если используется прерывание по захвату, то в процедуре обработки прерываний регистр ICRn 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 ICRn. После изменения фронта необходимо сбросить флаг захвата ICFn записью в него лог. 1. Если требуется измерение только частоты, то сброс флага ICFn не требуется (если используется обработка по прерыванию).

Блоки сравнения

16-разрядный цифровой компаратор непрерывно сравнивает значение TCNTn со значением регистра порога сравнения (OCRnx). Если значение TCNT равно OCRnx, то компаратор формирует сигнал совпадения (равенства). Следующий за совпадением такт ЦПУ устанавливает флаг

сравнения (OCFnx). Если бит OCIEnx = 1, то установка флага сравнения приведет к генерации прерывания по результату сравнения. Флаг OCFnx автоматически сбрасывается после перехода на вектор обработки прерывания. Альтернативно флаг OCFnx сбрасывается программно, если записать в него лог. 1. Сигнал совпадения используется формирователем выходного сигнала, результирующая форма которого зависит от выбранного с помощью бит WGMn3:0 режима работы таймера и режима формирования импульсов (биты COMnx1:0). 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”). 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 49 показана функциональная схема блока сравнения. Символ “n” в наименованиях бит и регистров заменяет номер таймера (1 и 3), а “x” заменяет наименование канала сравнения (A/B/C). Те блоки на функциональной схеме, которые не относятся к блоку сравнения, залиты серым цветом.

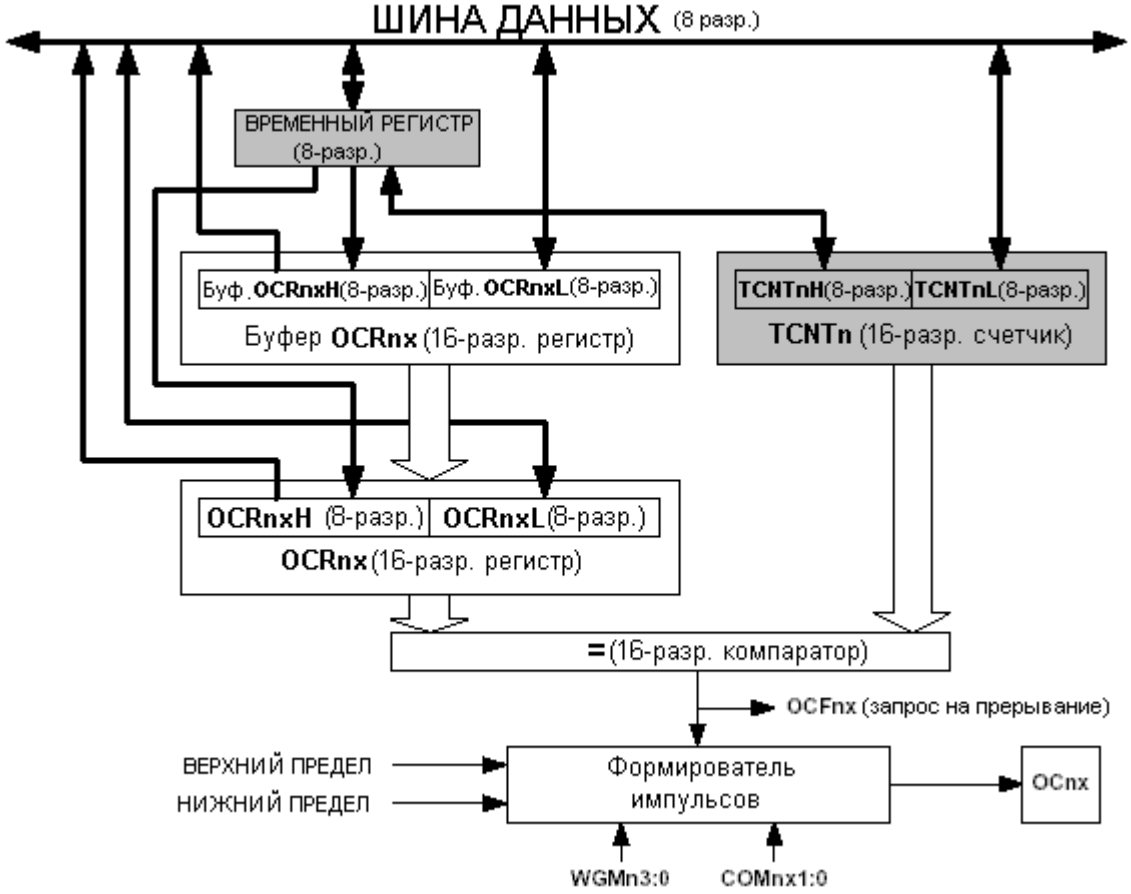


Рисунок 49 – Функциональная схема блока сравнения

Если задан любой из 12 режимов широтно-импульсной модуляции (ШИМ), то в этом случае регистр OCRnx содержит двойную буферизацию. Если таймер работает в нормальном режиме или режиме сброса при совпадении (СТС), то двойная буферизация отключается. Двойная буферизация синхронизирует обновление регистра порога сравнения OCRnx 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 OCRnx хоть и кажется сложным, но выполнен таким образом не случайно. Если двойная буферизация разрешена, то ЦПУ фактически осуществляет доступ к буферному регистру OCRnx. Если же двойная буферизация отключена, то ЦПУ обращается к регистру OCRnx 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 TCNTn и ICRn). Таким образом, OCRnx

считывается напрямую, а не через временный регистр старшего байта. Запись регистров OCRnx происходит через временный регистр, т.к. все 16 разр. участвуют в сравнении непрерывно. Первым необходимо записать старший байт OCRnxH. Если выполняется запись по адресу старшего байта, то содержимое временного регистра обновляется записываемым значением. Если выполняется запись младшего байта (OCRnxL), то параллельно копируется содержимое временного регистра в старшие 8-разрядов буферного регистра OCRnx или регистра порога сравнения OCRnx, тем самым обновляя все 16 разрядов за один такт ЦПУ (см. также "Доступ к 16-разр. регистрам").

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 FOCnx. Принудительная установка результата сравнения не приводит к установке флага OCFnx или сбросу/перезагрузке таймера, но влияет на состояние вывода OCnx, который будет устанавливаться, сбрасываться или переключаться (инвертироваться) в зависимости от выбранной установки бит COMnx.

Результат сравнения блокируется записью в TCNTn

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

Использование блока сравнения

Поскольку запись в TCNTn в любом из режимов работы блокирует отработку совпадения на один такт синхронизации таймера, то имеются некоторые опасные ситуации при изменении TCNTn во время использования любого из каналов сравнения, независимо работает таймер-счетчик или нет. Если в TCNTn записано значение равное OCRnx, то возникающее совпадение игнорируется, тем самым вызывая некорректную генерацию импульсов. Следует избегать записи в TCNTn 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 TCNTn значения равного нижнему пределу, если счетчик работает как вычитающий (обратный счет). Прежде чем настроить вывод OCnx на вывод в регистре направления данных необходимо выполнить инициализацию регистра OCnx. Самым простым способом решения этой задачи является использование бита принудительной установки результата сравнения (FOCnx) при работе таймера в нормальном режиме. Регистр OCnx сохраняет свое состояние даже при изменении режима работы таймера.

Учтите, что биты COMnx1:0 не содержат схемы двойной буферизации и любые их изменения вступают в силу мгновенно.

Блок формирования выходного сигнала

Биты задания режима формирования выходного сигнала (COMnx1:0) имеют двойное назначение. С одной стороны биты COMnx1:0 используются формирователем сигнала и определяют какое логическое состояние должно быть на выходе OCnx при возникновении следующего совпадения. С другой стороны, биты COMnx1:0 используются для разрешения/запрета альтернативной функции вывода порта OCnx. На рисунке 50 представлена упрощенная логическая схема, на которую воздействуют биты COMnx1:0. На рисунке показаны только те регистры управления портом ввода-вывода (DDR и PORT), на которые оказывает действие биты COMnx1:0. Если происходит системный сброс, то выход регистра OCnx принимает нулевое состояние.

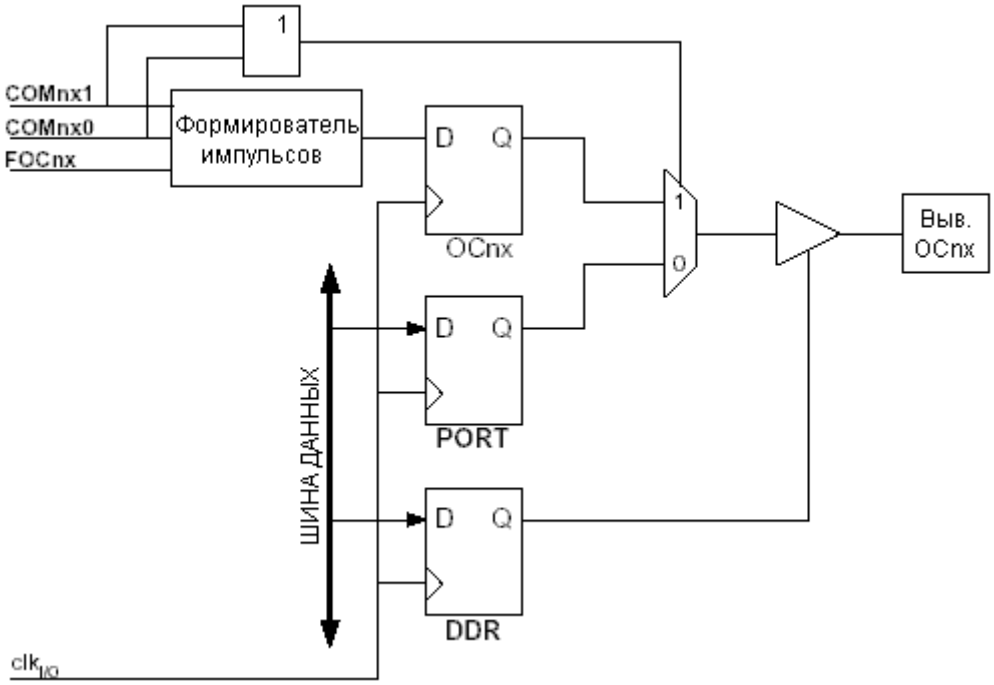


Рисунок 50 – Схема блока формирования выходного сигнала

Функция линии универсального порта ввода-вывода заменяется на функцию выхода формирователя сигнала OCnx, если хотя бы один из бит COM01, COM00 установлен (логика ИЛИ). Однако, управление направлением вывода OCnx (вход или выход) в этом случае остается за соответствующим битом регистра направления данных (DDR). Чтобы значение регистра OCnx присутствовало на выводе OCnx 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. 58 – 60).

Схемотехника выходной логики позволяет инициализировать состояние регистра OCnx перед разрешением настройки вывода OCnx в качестве выхода. Обратите внимание, что в некоторых режимах работы имеются зарезервированные состояния бит COMnx1:0. См. “Описание регистров 16-разр. таймеров-счетчиков”. Установки бит COMnx1:0 не оказывают никакого влияния на работу блока захвата.

Режимы генерации импульсов

Установки бит COMnx1:0 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 OCnx при возникновении совпадения, если COMnx1:0 = 0. В таблице 58 описано действие различных установок этих бит для режимов без ШИМ. Аналогичная информация для режима с быстрой ШИМ приведена в таблице 59, а для ШИМ с фазовой и частотной коррекцией в таблице 60.

Изменение состояния бит COMnx1:0 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 FOCnx.

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 (WGMn3-0) и режим формирования выходного сигнала (COMnx1:0). 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 COMnx1:0 позволяют включить/отключить инверсию на генерируемом ШИМ-выходе (т.е. выбрать ШИМ с инверсией или ШИМ без инверсии). Для режимов без ШИМ биты COMnx1:0 определяют, какое действие необходимо выполнить при возникновении совпадения: сбросить,

установить или инвертировать выход (см. также “**Блок формирования выходного сигнала**” и "**Временные диаграммы 16-разр. таймеров-счетчиков**").

Нормальный режим работы

Самым простым режимом работы является нормальный режим (WGMn3-0 = 0b0000). 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 TOVn будет установлен на том же такте синхронизации, когда TCNTn примет нулевое значение.

Фактически, флаг переполнения TOVn 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 TOVn 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 OCnx 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 (WGM01, WGM00 =0b10) регистр OCR0 используется для задания разрешающей способности счетчика. Если задан режим CTC и значение счетчика (TCNT0) совпадает со значением регистра OCR0, то счетчик обнуляется (TCNT0=0). Таким образом, OCR0 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.

В режиме сброса таймера при совпадении (WGMn3-0 = 0b0100 или 0b1100) разрешающая способность таймера задается регистрами OCRnA или ICRn. В режиме CTC происходит сброс счетчика (TCNTn), если его значение совпадает со значением регистра OCRnA (WGMn3-0 = 0b0100) или с ICRn (WGMn3-0 = 0b1100). Значение регистра OCRnA или ICRn 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 51. Счетчик (TCNTn) инкрементирует свое состояние до тех пор, пока не возникнет совпадение со значением OCRnA или ICRn, а затем счетчик (TCNTn) сбрасывается.

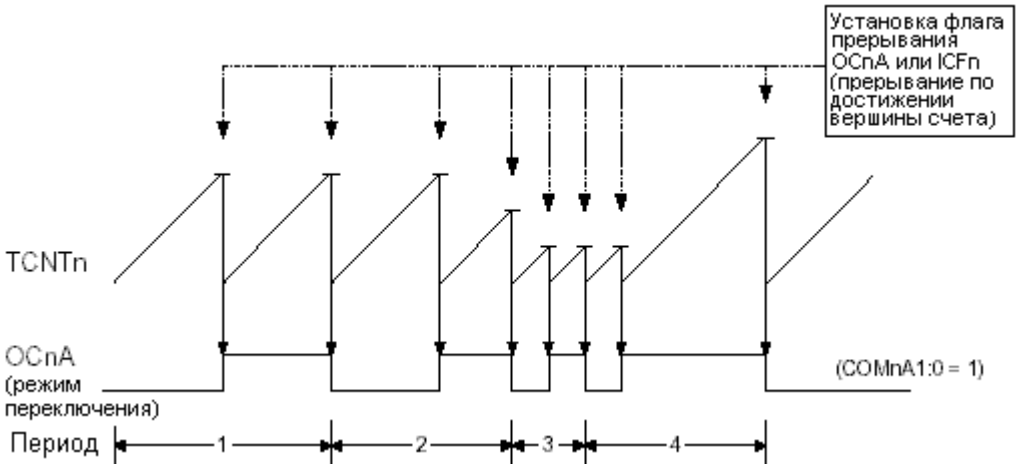


Рисунок 51 – Временная диаграмма для режима CTC

По достижении верхнего предела счета может генерироваться прерывание с помощью флагов OCFnA или ICFn, 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 OCRnA или ICRn, меньше текущего значения TCNTn, то сброс счетчика по условию совпадения наступит, когда он достигнет максимального значения (0xFFFF), затем перейдет в исходное состояние 0x0000 и достигнет нового значения OCRnA или ICRn. Во многих случаях возникновение такой ситуации не желательно. В качестве альтернативы может выступить режим быстрой ШИМ, где регистр OCRnA определяет верхний предел счета (WGMn3-0 = 0b1111), т.к. в этом случае OCRnA имеет двойную буферизацию.

Для генерации сигнала в режиме CTC выход OCnA может использоваться для изменения логического уровня при каждом совпадении, для чего необходимо задать режим переключения (COMnA1, COMnA0 = 0b01). Значение OCnA будет присутствовать на выводе порта, только если для данного вывода задано выходное направление. Максимальная частота генерируемого сигнала равна fOC0 = fclk_I/O/2, если OCRnA = 0x0000. Для других значений OCRn частоту генерируемого сигнала можно определить по формуле:

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

Также как и для нормального режима работы, флаг TOV0 устанавливается на том же такте таймера, когда его значение изменяется с 0xFFFF на 0x0000.

Режим быстрой ШИМ

Режим быстрой широтно-импульсной модуляции (ШИМ) (WGMn3-0 = 0b0101, 0b0110, 0b0111, 0b1110, 0b1111) 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 TCNTn и OCRnx сигнал OCnx устанавливается, а на верхнем пределе счета сбрасывается. Если задан инвертирующий режим, то выход OCnx 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 ICRn или OCRnA, но не менее 2 разрядов (ICRn или OCRnA = 0x0003) и не

более 16 разрядов (ICRn или OCRnA = 0xFFFF). Разрешающая способность ШИМ при заданном значении верхнего предела (ВП) вычисляется следующим образом:

В режиме быстрой ШИМ счетчик инкрементируется до совпадения его значения с одним из фиксированных значений 0x00FF, 0x01FF или 0x03FF (если WGMn3:0 = 0b0101, 0b0110 или 0b0111, соответственно), значением в ICRn (если WGMn3:0 = 0b1110) или значением в OCRnA (если WGMn3:0 = 0b1111), а затем сбрасывается следующим тактом синхронизации таймера. Временная диаграмма для режима быстрой ШИМ представлена на рисунке 52. На рисунке показан режим быстрой ШИМ, когда для задания верхнего предела используется регистр OCRnA или ICRn. Значение TCNTn 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 TCNTn, где совпадают значения OCRnx и TCNTnx. Флаг прерывания OCnx устанавливается при возникновении совпадении.

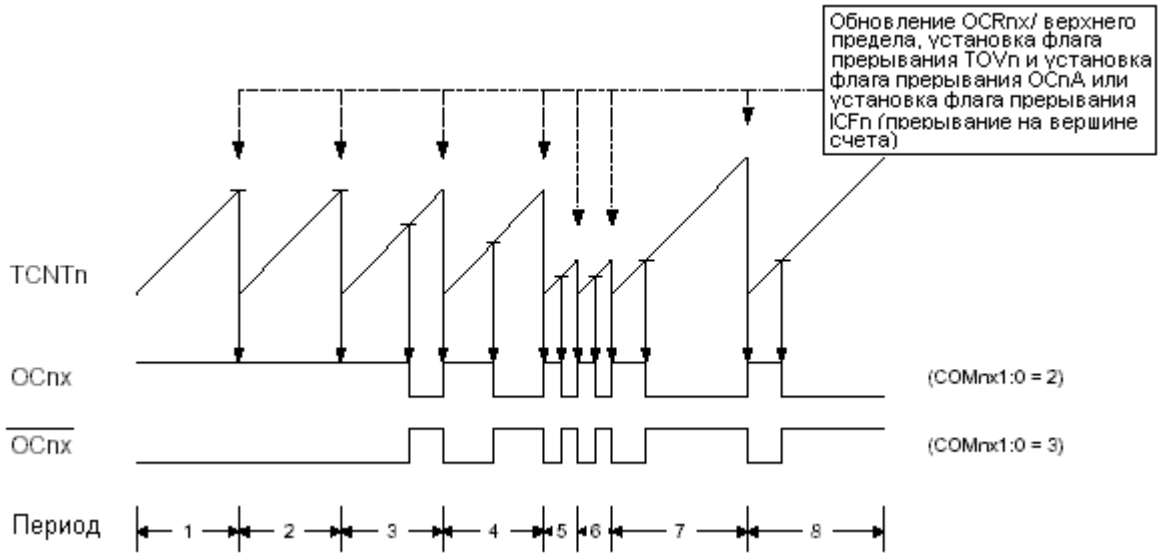


Рисунок 52 – Временная диаграмма для режима быстрой ШИМ

Флаг переполнения таймера-счетчика (TOVn) устанавливается всякий раз, когда счетчик достигает верхнего предела. Дополнительно тем же тактовым импульсом вместе с флагом TOVn могут установиться флаги OCnA или ICFn, если для задания верхнего предела используется регистр OCRnA или ICRn, 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 TCNTn и OCRnx никогда не возникнет. Обратите внимание, что при использовании фиксированных значений верхнего предела во время записи в регистры OCRnx происходит маскирование к 0 неиспользуемых разрядов.

Механизм модификации регистра ICRn отличается от OCRnA в том случае, если он используется для задания верхнего предела. Регистр ICRn не имеет двойной буферизации. Это означает, что если в ICRn записывается малое значение во время работы счетчика с малым предделением или без него, то имеется опасность записи в регистр ICRn значения, которое окажется меньше текущего значения TCNTn. 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 OCRnA содержит схему двойной буферизации, поэтому, его можно модифицировать в любой момент времени.

Если выполняется запись по адресу OCRnA, то фактически значение помещается в буферный регистр OCRnA. Если же возникает совпадение между TCNTn и вершиной счета, то следующим тактом синхронизации таймера происходит копирование буферного регистра в регистр порога

сравнения OCRnA. Обновление регистра выполняется тем же тактом, что и сброс TCNTn и установка флага TOVn.

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

В режиме быстрой ШИМ блоки сравнения позволяют генерировать ШИМ-сигналы на выводах OCnx. Если COMnx1:0 =0b10, то задается ШИМ без инверсии выхода, а если COMnx1:0 = 0b11, то задается режим ШИМ с инверсией на выходе (см. таблицу 59). Фактическое значение OCnx можно наблюдать на выводе порта, если для него задано выходное направление (DDR_OCnx). ШИМ-сигнал генерируется путем установки (сброса) регистра OCnx при возникновении совпадения между OCRnx и TCNTn, а также путем сброса (установки) регистра OCnx вместе со сбросом счетчика (переход с верхнего предела на нижний предел).

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

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

Запись предельных значений в регистр OCRnx связана с особыми случаями в генерации ШИМ-импульсов. Если OCRnx установить равным нижнему пределу (0x0000), то на выходе будет возникать короткий импульс каждый (ВП+1)-ый такт синхронизации таймера. Запись в OCRnx значения равного верхнему пределу приведет к установке постоянного уровня лог. 1 или 0 на выходе (зависит от выбранной с помощью бит COMnx1:0 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 COMnA1:0 = 0b01, которая вызывает переключение (инвертирование) логического уровня на выходе OCnA при каждом совпадении. Данное применимо, только если OCRnA используется для задания верхнего предела (WGMn3-0 =0b1111). Максимальная генерируемая частота меандра в этом случае fOCnA = fclk_I/O/2, если OCRnA =0x0000. Данная особенность аналогична переключению OCnA 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) (WGMn3-0 = 0b0001, 0b010, 0b0011, 0b1010 или 0b1011) 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 OCnx сбрасывается/устанавливается при совпадении значений TCNTn и OCRnx во время прямого/обратного счета. Если задан инвертирующий режим выхода, то, наоборот, во время прямого счета происходит установка, а во время обратного – сброс выхода OCnx. 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 ICRn или OCRnA. Минимальная разрешающая способность равна 2-м разрядам (ICRn или OCRnA = 0x0003), а максимальная -16-ти разрядам (ICRn или OCRnA =0xFFFF). Если задан верхний предел, то разрешающая способность ШИМ в данном режиме определяется следующим образом:

В режиме ШИМ ФК счетчик инкрементируется пока не достигнет одного из фиксированных значений 0x00FF, 0x01FF или 0x03FF (соответственно для WGMn3-0 = 0b0001, 0b0010 или 0b0011), а также значения равного ICRn (если WGMn3-0 = 0b1010) или OCRnA (если WGMn3:0 = 0b1011). Далее, при достижении верхнего предела, счетчик изменяет направление счета. Значение TCNTn остается равным верхнему пределу в течение одного такта синхронизации таймера. Временная диаграмма для режима ШИМ ФК представлена на рисунке 53. На рисунке показан режим ШИМ ФК с использованием регистра OCRnA или ICRn для задания верхнего предела. Состояние TCNTn 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 TCNTn, где возникает совпадение со значением OCRnx. Флаг прерывания OCnx устанавливается при возникновении совпадения.

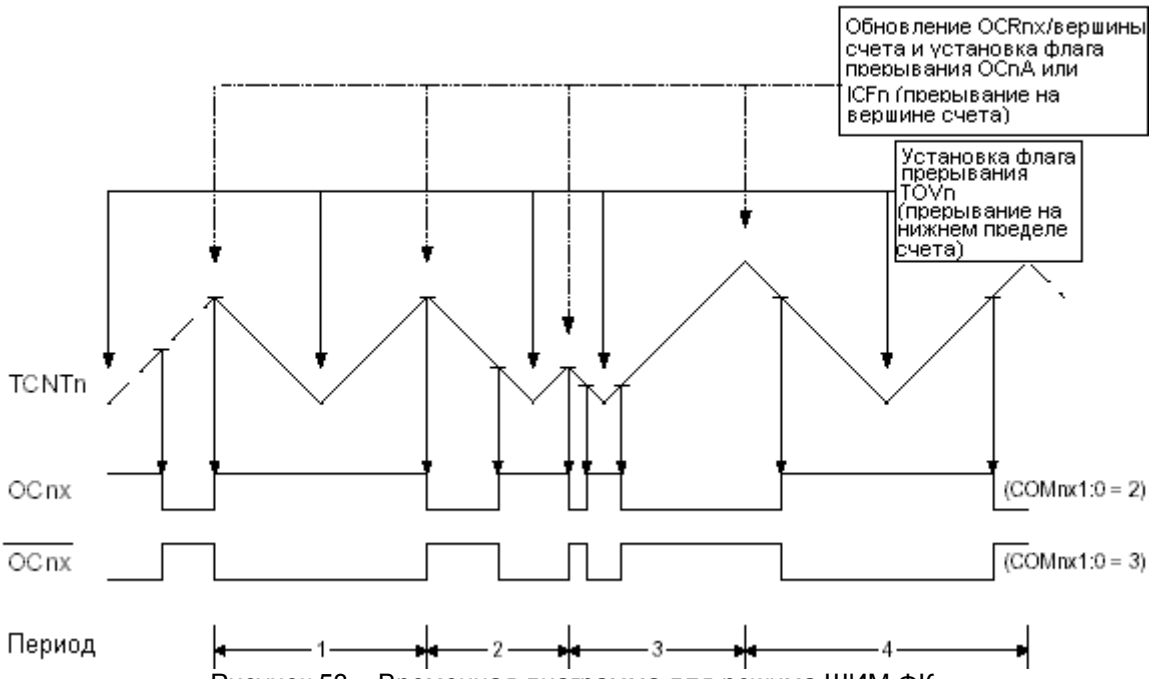


Рисунок 53 – Временная диаграмма для режима ШИМ ФК

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

При изменении значения верхнего предела счета необходимо следить, чтобы оно было больше или равно значениям во всех регистрах сравнения. В противном случае совпадение между TCNTn и OCRnx никогда не возникнет. Обратите внимание, что при использовании фиксированных значений верхнего предела счета во время записи в регистры OCRnx неиспользуемые разряды обнуляются. Третий период на рисунке 53 иллюстрирует случай, когда динамическое изменение верхнего предела счета приводит к генерации несимметричного импульса. Данная особенность основывается на времени обновления регистра OCRnx. Поскольку, обновление OCRnx 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 OCnx. Если установить COMnx1:0 = 0b10, то выход ШИМ будет без инверсии, а если COMnx1:0=0b11, то с инверсией (см. таблицу 60). Фактическое значение OCnx можно наблюдать на выводе порта, если в регистре направления данных для данного вывода порта задано выходное направление (DDR_OCnx). ШИМ-сигнал генерируется путем установки (сброса) регистра OCnx при совпадении значений OCRnx и TCNTn во время прямого счета, а также путем сброса (установки) регистра OCnx при совпадении между OCRnx и TCNTn во время обратного счета. Результирующая частота ШИМ-сигнала в режиме ШИМ ФК при заданном верхнем пределе (ВП) может быть вычислена по следующему выражению:

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

Запись предельных значений в регистр OCRnx связано с особыми случаями в генерации ШИМ-сигналов в режиме ШИМ ФК. Если задать режим ШИМ без инверсии и OCRnx 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 OCnA в качестве верхнего предела (WGMn3:0 = 0b1011) и установить COMnA1:0 =0b01, то на выходе OCnA будет генерироваться меандр.

Режим широтно-импульсной модуляции с фазовой и частотной коррекцией

Режим широтно-импульсной модуляции с фазовой и частотной коррекцией (ШИМ ФЧК) (WGMn3-0 = 0b1000 или 0b1001) 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 OCnx сбрасывается, если возникает совпадение между TCNTn и OCRnx 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 OCRnx из буферного регистра OCRnx (см. рисунок 53 и рисунок 54).

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

В режиме ШИМ ФЧК счетчик инкрементируется до совпадения со значением в ICRn (WGMn3:0 = 0b1000) или в OCRnA (WGMn3:0 = 0b1001). Это означает достижение вершины счета, после чего происходит изменение направления счета. Значение TCNTn остается равным вершине счета в течение одного такта синхронизации таймера. Временная диаграмма для режима ШИМ ФЧК показана на рисунке 54. На рисунке показан режим ШИМ ФЧК, когда вершину счета задает регистр OCRnA или ICRn. Значение TCNTn 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 TCNTn, где возникает совпадение между OCRnx и TCNTn. Флаг прерывания OCnx устанавливается после возникновения совпадения.

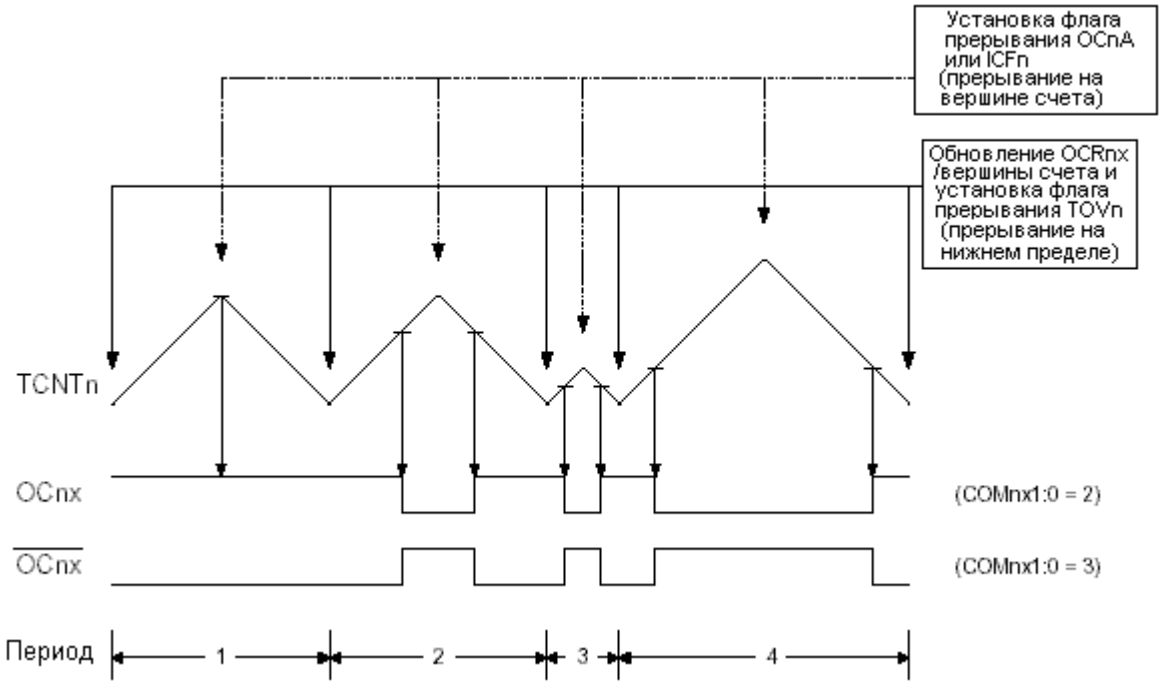


Рисунок 54 – Временная диаграмма режима ШИМ с фазовой и частотной коррекцией

Флаг переполнения таймера-счетчика (TOVn) устанавливается тем же тактом, когда произошло обновление регистров значением из буферного регистра (на нижнем пределе счета). Если для задания верхнего предела используется регистр OCRnA или ICRn, то по достижении счетчиком верхнего предела устанавливается флаг OCnA или ICFn, 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 TCNTn и OCRnx никогда не наступит.

На рисунке 54 показано, что в отличие от режима ШИМ ФК, генерируемый выходной сигнал симметричен на всех периодах. Поскольку, регистры OCRnx 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 ICRn для задания верхнего предела рекомендуется, если значение верхнего предела является константой. В этом случае также освобождается регистр OCRnA для широтно-импульсной модуляции импульсов на выводе OCnA. Однако если требуется динамическое изменение частоты ШИМ за счет изменения верхнего предела, то для задания верхнего предела рекомендуется использовать регистр OCRnA за счет наличия у него двойной буферизации.

В режиме ШИМ ФЧК блоки сравнения позволяют генерировать ШИМ-импульсы на выводе OCnx. Если COMnx1:0 = 0b10, то задается неинвертирующий ШИМ выход, а, если COMnx1:0=0b11, то инвертирующий (см. таблицу 60). Значение OCnx будет присутствовать на соответствующем выводе порта только в случае, если для него задано выходное направление. ШИМ сигнал генерируется путем установки (сброса) регистра OCnx при совпадении между OCRnx и TCNTn во время прямого счета и сброса (установки) регистра OCnx при совпадении между OCRnx и TCNTn во время обратного счета. Частота ШИМ в данном режиме при заданном верхнем пределе (ВП) счета определяется следующим образом:

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

Запись предельных значений в регистр OCRnx связана с особыми случаями в генерации ШИМ-сигналов в данном режиме. Если задать OCRnx 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 OCRnA используется для задания верхнего предела (WGMn3:0 = 0b1001) и COMnA1:0 = 0b01, то на выходе OCnA будет генерироваться меандр.

Временные диаграммы 16-разр. таймеров-счетчиков

16-разр. таймеры выполнены по синхронной схеме, поэтому, сигнал синхронизации таймера (clkTn) на следующих рисунках показан как разрешающий сигнал. На рисунках также представлена информация по моментам установки флагов прерываний и обновления регистра OCRnx значением из буферного регистра OCRnx (только для режимов с двойной буферизацией). На рисунке 55 представлена временная диаграмма с установкой флага OCFnx. На рисунке 56 представлена аналогичная диаграмма, но с разрешенным предделением. На рисунке 57 иллюстрируется алгоритм счета в районе верхнего предела в различных режимах. Если используется режим ШИМ ФЧК, то регистр OCRnx 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 TOVn устанавливается на нижнем пределе. На рисунке 58 приведена аналогичная предыдущей временная диаграмма, но с разрешенным предделением.

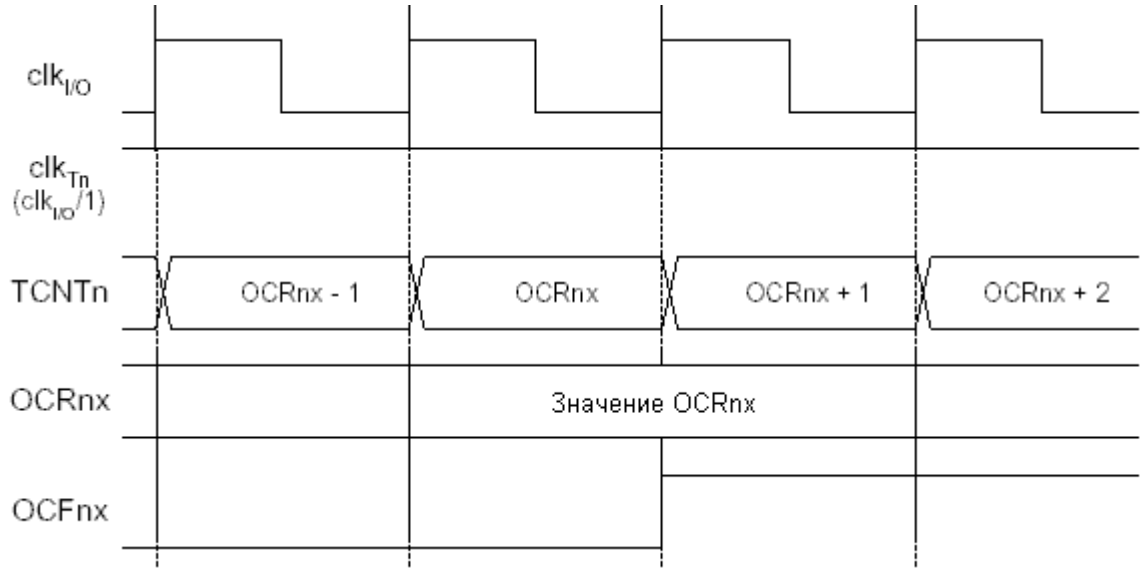


Рисунок 55 – Временная диаграмма таймера-счетчика без предделения с установкой OCFnx

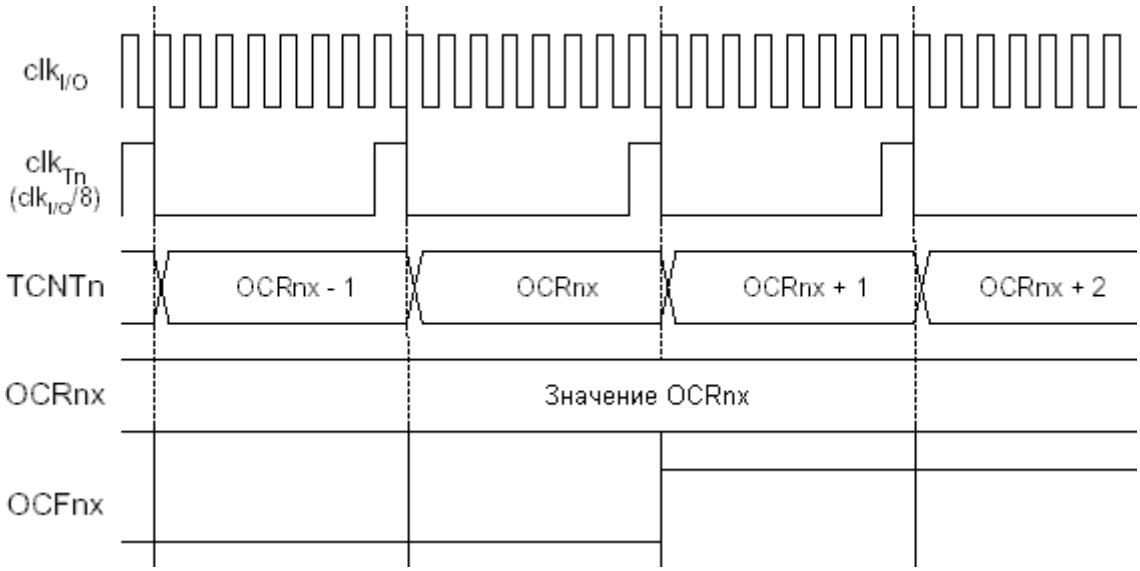


Рисунок 56 – Временная диаграмма таймера-счетчика с предделением на 8 (fclk_I/O/8) и установкой OCFnx

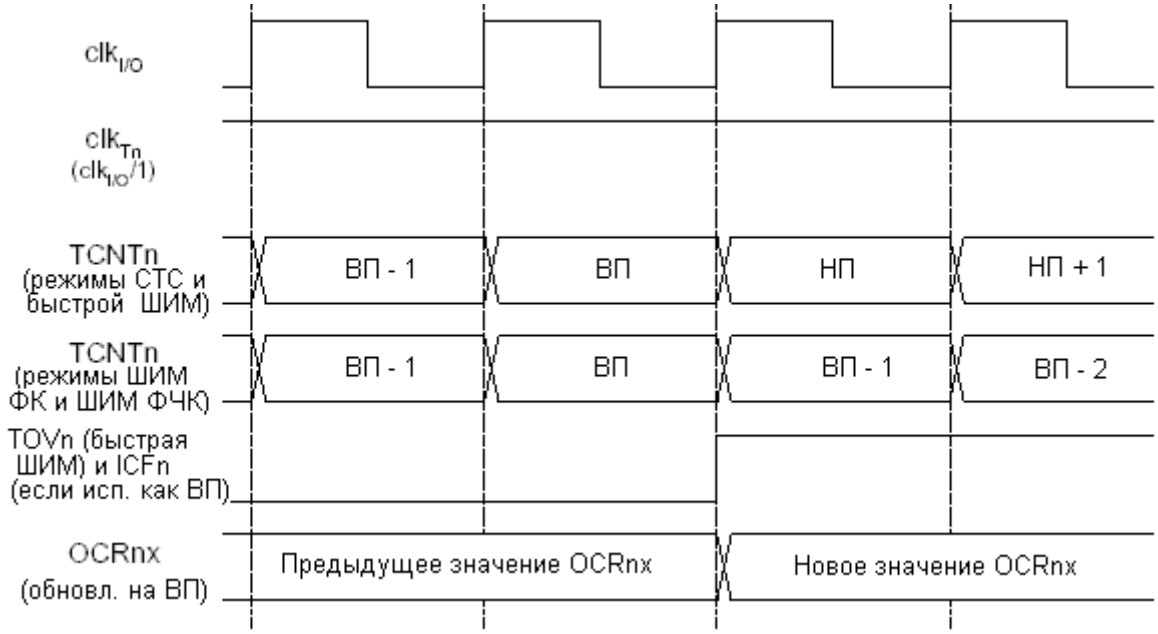


Рисунок 57 – Временная диаграмма таймера-счетчика без предделения

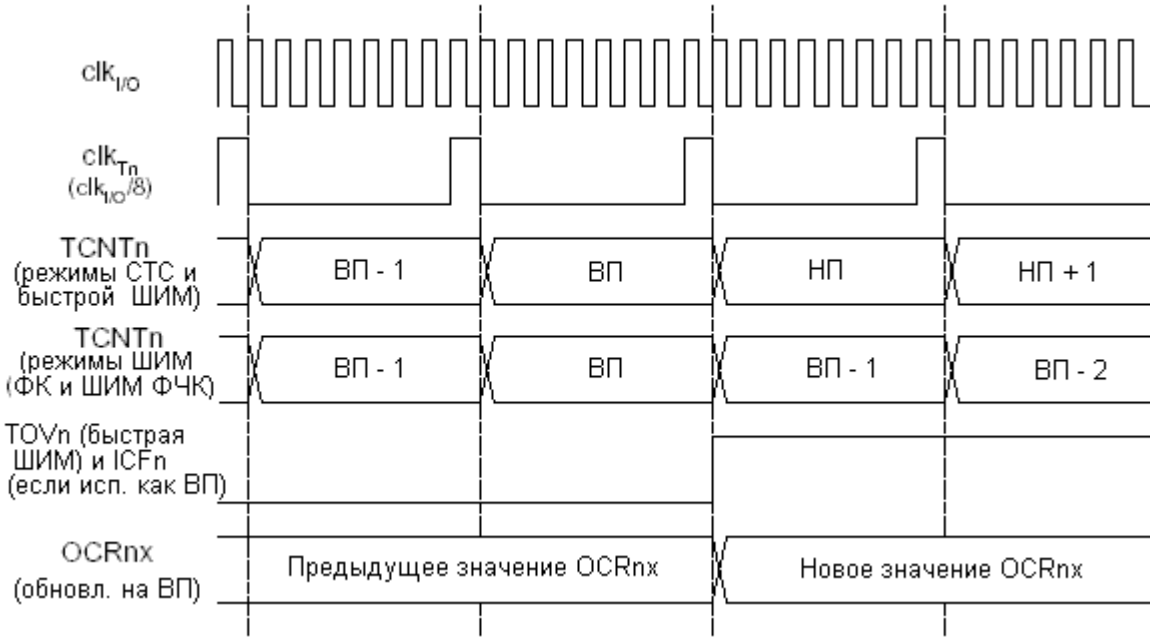


Рисунок 58 - Временная диаграмма таймера-счетчика с предделением на 8 (fclk_I/O/8)

Описание регистров 16-разр. таймеров-счетчиков

Регистр A управления таймером-счетчиком 1 – TCCR1A

Разряд	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	**TCCR1A**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр A управления таймером-счетчиком 3 – TCCR3A

Разряд	7	6	5	4	3	2	1	0	
	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	**TCCR3A**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряды 7:6 – COMnA1:0: Режим формирования выходного сигнала канала A

Разряды 5:4 – COMnB1:0: Режим формирования выходного сигнала канала B

Разряды 3:2 – COMnC1:0: Режим формирования выходного сигнала канала C

Биты COMnA1:0, COMnB1:0 и COMnC1:0 влияют на работу выводов OCnA, OCnB и OCnC, соответственно. Если один или оба бита COMnA1:0 равны 1, то вывод OCnA переходит к выполнению альтернативной функции, запрещая его работу как обычного порта ввода-вывода. Аналогичные изменения происходят с выводами OCnB и OCnC во время записи лог. 1 в один из битов COMnB1:0 и COMnC1:0, соответственно. 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 OCnA, OCnB или OCnC.

Если выбрано подключение сигналов OCnA, OCnB или OCnC к выводам микроконтроллера, то назначение бит COMnx1:0 определяется выбранным с помощью бит WGMn3:0 режима работы

таймера-счетчика. В таблице 58 показано назначение бит COMnx1:0, когда битами WGMn3:0 выбран режим сброса при совпадении (CTC) или нормальный режим, т.е. режимы без ШИМ.

Таблица 58 – Режимы формирования выходного сигнала в режимах работы таймера без ШИМ

COMnA1/COMnB1/COMnC1	COMnA0/COMnB0/COMnC0	Описание
0	0	Нормальная работа порта, сигналы OCnA/OCnB/OCnC отключены.
0	1	Переключение (инвертирование) OCnA/OCnB/OCnC при совпадении.
1	0	Сброс OCnA/OCnB/OCnC при совпадении (установка лог. 0).
1	1	Установка OCnA/OCnB/OCnC при совпадении (установка лог. 1).

В таблице 59 представлено назначение бит COMnx1:0, когда с помощью бит WGMn3:0 выбран режим быстрой ШИМ.

Таблица 59 – Режим формирования выходного сигнала в режиме работа таймера с быстрой ШИМ

COMnA1/COMnB1/COMnC1	COMnA0/COMnB0/COMnC0	Описание
0	0	Нормальная работа порта, сигналы OCnA/OCnB/OCnC отключены.
0	1	WGMn3:0 = 15: Переключение (инвертирование) OCnA при совпадении, OCnB/OCnC отключены (нормальная работа порта). Для всех других установок WGMn соответствует нормальная работа порта, когда OCnA/OCnB/OCnC отключены.
1	0	Сброс OCnA/OCnB/OCnC при совпадении, установка OCnA/OCnB/OCnC на вершине счета
1	1	Установка OCnA/OCnB/OCnC при совпадении, сброс OCnA/OCnB/OCnC на вершине

Прим.: Имеются особые случаи, когда OCRnA/OCRnB/OCRnC равно верхнему пределу счета и установлен COMnA1/COMnB1/COMnC1. В этом случае возникшее совпадение игнорируется, но установка или сброс на вершине счета выполняется (см. “Режим быстрой ШИМ”).

В таблице 59 представлено назначение бит COMnx1:0 для режима ШИМ ФК и ШИМ ФЧК

Таблица 60 – Режим формирования выходного сигнала в режимах работы таймера с ШИМ ФК и ШИМ ФЧК

COMnA1/COMnB1/COMnC1	COMnA0/COMnB0/COMnC0	Описание
0	0	Нормальная работа порта, сигналы OCnA/OCnB/OCnC отключены.
0	1	WGMn3:0 = 9 или 14: Переключение (инвертирование) OCnA при совпадении, OCnB/OCnC отключены (нормальная работа порта). Для всех других

		установок WGMn соответствует нормальная работа порта, когда OCnA/OCnB/OCnC отключены.
1	0	Сброс OCnA/OCnB/OCnC при совпадении во время прямого счета, установка OCnA/OCnB/OCnC при совпадении во время обратного счета
1	1	Установка OCnA/OCnB/OCnC при совпадении во время прямого счета, сброс OCnA/OCnB/OCnC при совпадении во время обратного счета

Прим.: Имеются особые случаи, когда OCRnA/OCRnB/OCRnC равно верхнему пределу и установлен COMnA1/COMnB1//COMnC1 (см. “Режим ШИМ с фазовой коррекцией”).

Разряд 1:0 – WGMn1:0: Режим работы таймера-счетчика

В сочетании с битами WGMn3:2 из регистра TCCRnB данные биты определяют алгоритм счета, источник для задания вершины счета (ВП) и тип генерируемой формы сигнала (см. табл. 61). 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").

Таблица 61 – Режимы работы таймера-счетчика

Режим	**WGMn3**	**WGMn2 (CTCn)**	**WGMn1 (PWMn1)**	**WGMn0 (PWMn0)**	**Режим работа таймера-счетчика(1)**	**Верхний предел**	**Обновление OCRnx**	**Установка флага TOVn 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	OCRnA	сразу после записи	МАКС
5	0	1	0	1	8-разр. быстрая ШИМ	0x00FF	на вершине счета	на вершине счета
6	0	1	1	0	9-разр. быстрая ШИМ	0x01FF	на вершине счета	на вершине счета
7	0	1	1	1	10-разр. быстрая ШИМ	0x03FF	на вершине счета	на вершине счета
8	1	0	0	0	ШИМ ФЧК	ICRn	на нижнем пределе	нижнем пределе
9	1	0	0	1	ШИМ ФЧК	OCRnA	на нижнем пределе	нижнем пределе
10	1	0	1	0	ШИМ ФК	ICRn	на вершине счета	нижнем пределе
11	1	0	1	1	ШИМ ФК	OCRnA	на вершине счета	нижнем пределе
12	1	1	0	0	СТС	ICRn	сразу после записи	МАКС.

13	1	1	0	1	(резерв)	–	-	-
14	1	1	1	0	Быстрая ШИМ	ICRn	на вершине счета	на вершине счета
15	1	1	1	1	Быстрая ШИМ	OCRnA	на вершине счета	на вершине счета

Прим.: 1. Наименования бит CTCn и PWMn1:0 являются устаревшими, поэтому, необходимо использовать имена WGMn2:0. Однако назначение и расположение этих бит совместимо с предыдущими версиями таймеров.

Регистр B управления таймером-счетчиком 1 – TCCR1B

Разряд	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	**TCCR1B**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр B управления таймером-счетчиком 3 – TCCR3B

Разряд	7	6	5	4	3	2	1	0	
	ICNC3	ICES3	-	WGM33	WGM32	CS32	CS31	CS30	**TCCR3B**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 7 – ICNCn: Подавитель шума на входе захвата

Установка данного бита (запись лог. 1) активизирует подавитель шума на входе захвата. После активизации подавителя шумов сигнал с вывода ICPn 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.

Разряд 6 – ICESn: Выбор детектируемого фронта на входе захвата

Данный бит позволяет задать, какой фронт на входе захвата ICPn приведет к захвату состояния таймера. Если ICESn =0, то падающий (отрицательный) фронт приводит к захвату состояния таймера, а если же ICESn = 1, то нарастающий (положительный) фронт приводит к возникновению захвата.

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

Если регистр ICRn используется для хранения значения верхнего предела счета (см. табл. 61), то вход ICPn отключается от соответствующего вывода микроконтроллера и функция захвата блокируется.

Разряд 5 – Зарезервированный бит

Данный бит зарезервирован для дальнейшего использования. В целях совместимости с будущими разработками рекомендуется во время записи в регистр TCCRnB в данном разряде указывать лог. 0.

Разряд 4:3 – WGMn3:2: Режим работы таймера-счетчика

См. описание регистр TCCRnA.

Разряд 2:0 – CSn2:0: Выбор тактового источника

Данный три бита позволяют выбрать тактовый источник для таймера-счетчика (см. рисунок 55 и рисунок 56).

Таблица 62 – 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 Tn, то данная функция за ним сохраняется, даже при его настройке на вывод. Данная функция позволяет программно управлять счетом.

Регистр C управления таймером-счетчиком 1 – TCCR1C

Разряд	7	6	5	4	3	2	1	0	
	FOC1A	FOC1B	FOC1C	-	-	-	-	-	**TCCR1C**
Чтение/запись	Зп.	Зп.	Зп.	Чт.	Чт.	Чт.	Чт.	Чт.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр C управления таймером-счетчиком 3 – TCCR3C

Разряд	7	6	5	4	3	2	1	0	
	FOC3A	FOC3B	FOC3C	-	-	-	-	-	**TCCR3C**
Чтение/запись	Зп.	Зп.	Зп.	Чт.	Чт.	Чт.	Чт.	Чт.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 7 – FOCnA: Бит принудительной установки результата сравнения для канала A

Разряд 6 – FOCnB: Бит принудительной установки результата сравнения для канала B

Разряд 5 – FOCnC: Бит принудительной установки результата сравнения для канала C

Биты FOCnA/FOCnB/FOCnC становятся активными, когда с помощью бит WGMn3:0 выбран режим без ШИМ. В этом случае запись в FOCnA/FOCnB/FOCnC лог. 1 приводит к немедленной установке результата сравнения на входе блока формирователя сигнала. Выход OCnA/OCnB/OCnC изменяется в соответствии с установками бит COMnx1:0. Обратите внимание, что биты FOCnA/FOCnB/FOCnC реализованы как стробы. Стробы FOCnA/FOCnB/FOCnC не генерируют каких-либо прерываний и сбрасывают счетчик в режиме сброса таймера при совпадении (CTC), где OCRnA используется для задания вершины счета.

При чтении бит FOCnA/FOCnB/FOCnB всегда возвращается нулевое значение.

Разряды 4:0 – Зарезервированные биты

Данные биты зарезервированы для дальнейшего использования. В целях совместимости с последующими разработками во время записи в регистр TCCRnC необходимо записывать лог. 0.

Таймер-счетчик 1 – TCNT1H и TCNT1L

Разряд	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								**TCNT1H**
	TCNT1[7:0]								**TCNT1L**
Чтение/запись	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. значение	0	0	0	0	0	0	0	0	

Таймер-счетчик 3 – TCNT3H и TCNT3L

Разряд	7	6	5	4	3	2	1	0	
	TCNT3[15:8]								**TCNT3H**
	TCNT3[7:0]								**TCNT3L**
Чтение/запись	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. значение	0	0	0	0	0	0	0	0	

Две ячейки в области ввода-вывода (TCNTnH и TCNTnL, вместе TCNTn) 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”).

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

Регистр сравнения 1 A – OCR1AH и OCR1AL

Разряд	7	6	5	4	3	2	1	0	
	OCR1A [15:8]								**OCR1AH**
	OCR1A [7:0]								**OCR1AL**
Чтение/запись	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр сравнения 1 B – OCR1BH и OCR1BL

Разряд	7	6	5	4	3	2	1	0	
	OCR1B [15:8]								**OCR1BH**
	OCR1B [7:0]								**OCR1BL**
Чтение/запись	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр сравнения 1 C – OCR1CH и OCR1CL

Разряд	7	6	5	4	3	2	1	0	
	OCR1C [15:8]								**OCR1CH**
	OCR1C [7:0]								**OCR1CL**
Чтение/запись	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр сравнения 3 A – OCR3AH и OCR3AL

Разряд	7	6	5	4	3	2	1	0	
	OCR3A [15:8]								**OCR3AH**
	OCR3A [7:0]								**OCR3AL**
Чтение/запись	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр сравнения 3 B – OCR3BH и OCR3BL

Разряд	7	6	5	4	3	2	1	0	
	OCR3B [15:8]								**OCR3BH**
	OCR3B [7:0]								**OCR3BL**
Чтение/запись	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр сравнения 3 C – OCR3CH и OCR3CL

Разряд	7	6	5	4	3	2	1	0	
	OCR3C [15:8]								**OCR3CH**
	OCR3C [7:0]								**OCR3CL**
Чтение/запись	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. значение	0	0	0	0	0	0	0	0	

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

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”).

Регистр захвата 1 – ICR1H и ICR1L

Разряд	7	6	5	4	3	2	1	0	
	ICR1C [15:8]								**ICR1CH**
	ICR1C [7:0]								**ICR1CL**
Чтение/запись	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр захвата 3 – ICR3H и ICR3L

Разряд	7	6	5	4	3	2	1	0	
	ICR3C [15:8]								**ICR3CH**
	ICR3C [7:0]								**ICR3CL**
Чтение/запись	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистры захвата обновляются содержимым соответствующего счетчика (TCNTn) при каждом определении условия захвата на входе ICPn (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”).

Регистр маски прерываний таймера-счетчика – TIMSK

Разряд	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	**TIMSK**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

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.

Разряд 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 в регистре TIFR, программа переходит на соответствующий вектор прерывания (см. раздел "Прерывания").

Разряд 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").

Разряд 3 – OCIE1B: Разрешение прерывания по результату сравнения канала B таймера-счетчика 1

Действие аналогично предыдущему, но в отношении канала сравнения B.

Разряд 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.

Расширенный регистр маски прерываний таймера-счетчика– ETIMSK

Разряд	7	6	5	4	3	2	1	0	
	-	-	TICIE3	OCIE3A	OCIE3B	TOIE3	OCIE3C	TOIE3C	**ETIMSK**
Чтение/запись	Чт.	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Прим.: Данный регистр не доступен в режиме совместимости с ATmega103.

Разряд 7:6 – Зарезервированные биты

Данные биты зарезервированы для дальнейшего использования. В целях совместимости с последующими версиями во время записи в регистр в данный бит необходимо записывать лог. 0.

Разряд 5 – TICIE3: Разрешение прерывания по захвату состояния таймера-счетчика 3

Если в данный бит записана лог. 1 и установлен флаг I в регистре статуса, то разрешается работа прерывания по захвату состояния таймера-счетчика 3. Если устанавливается флаг ICF3 в регистре ETIFR, то программа переходит на соответствующий вектор прерывания (см. раздел "Прерывания").

Разряд 4 – OCIE3A: Разрешение прерывания по результату сравнения канала A таймера-счетчика 3

Если данный бит равен 1 и установлен флаг I в регистре статуса, то разрешается работа прерывания по результату сравнения канала A таймера-счетчика 3. Если устанавливается флаг OCF3A в регистре ETIFR, то программа переходит на соответствующий вектор прерывания (см. раздел "Прерывания").

Разряд 3 – OCIE3B: Разрешение прерывания по результату сравнения канала B таймера-счетчика 3

Аналогично предыдущему, но по отношению к каналу B.

Разряд 2 – TOIE3: Разрешение прерывания по переполнению таймера-счетчика

Если в данный бит записана лог. 1 и установлен флаг I в регистре статуса, то разрешается прерывание по переполнению таймера-счетчика 3. После этого, установка флага TOV3 в регистре ETIFR приведет к переходу на соответствующий вектор прерывания (см. раздел "Прерывания").

Разряд 1 – OCIE3C: Разрешение прерывания по результату сравнения канала C таймера-счетчика 3

Если данный бит равен 1 и установлен флаг I в регистре статуса, то разрешается работа прерывания по результату сравнения канала C таймера-счетчика 3. Если устанавливается флаг OCF3C в регистре ETIFR, то программа переходит на соответствующий вектор прерывания (см. раздел "Прерывания").

Разряд 0 – OCIE3C: Разрешение прерывания по результату сравнения канала C таймера-счетчика 1

Если данный бит равен 1 и установлен флаг I в регистре статуса, то разрешается работа прерывания по результату сравнения канала C таймера-счетчика 1. Если устанавливается флаг OCF1C в регистре ETIFR, то программа переходит на соответствующий вектор прерывания (см. раздел "Прерывания").

Регистр флагов прерываний таймеров-счетчиков – TIFR

Разряд	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOF0	**TIFR**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	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.

Разряд 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 WGMn3:0 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.

Разряд 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.

Разряд 3 – OCF1B: Флаг результата сравнения канала B таймера-счетчика 1

Данный флаг действует аналогично предыдущему, но в отношении канала сравнения B.

Разряд 2 – TOV1: Флаг переполнения таймера-счетчика 1

Установка данного флага зависит от значений бит WGMn3:0. В нормальном режиме и режиме CTC флаг TOV1 устанавливается при переполнении таймера-счетчика. См. табл. 61 для изучения поведения флага TOV1 при задании других значений WGMn3:0. Флаг TOV1 автоматически сбрасывается при переходе на вектор прерывания по переполнению таймера-счетчика 1. Альтернативно флаг TOV1 сбрасывается путем записи в него лог. 1.

Расширенный регистр флагов прерываний таймеров-счетчиков – ETIFR

Разряд	7	6	5	4	3	2	1	0	
	-	-	ICF3	OCF3A	OCF3B	TOV3	OCF3C	OCF1C	**ETIFR**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряды 7:6 – Зарезервированные биты

Данные биты зарезервированы для дальнейшего использования. В целях совместимости с последующими версиями необходимо в данные разряды записывать лог. 0 во время записи в регистр ETIFR.

Разряд 5 – ICF3: Флаг захвата состояния таймера-счетчика 3

Флаг устанавливается, если на входе ICP3 определяется условие захвата. Если регистр захвата ICR3 выбран с помощью бит WGM3:0 в качестве источника верхнего предела счета, то флаг ICF3 устанавливается по достижении верхнего предела счета.

ICF3 автоматически сбрасывается при переходе на вектор прерывания по захвату состояния таймера-счетчика. Альтернативно флаг ICF3 можно сбросить путем записи в него лог. 1.

Разряд 4 – OCF3A: Флаг результата сравнения канала A таймера-счетчика 3

Данный флаг устанавливается следующим тактом после совпадения значения TCNT3 с регистром A порога сравнения (OCR3A).

Обратите внимание, что строб принудительной установки результата сравнения (FOC3A) не устанавливает флаг OCF3A. Флаг OCF3A автоматически сбрасывается при переходе на соответствующий вектор прерывания. Альтернативно, флаг OCF3A сбрасывается путем записи в него лог. 1.

Разряд 3 – OCF3B: Флаг результата сравнения канала B таймера-счетчика 3

Действует аналогично предыдущему, но в отношении канала B таймера-счетчика 3.

Разряд 2 – TOV3: Флаг переполнения таймера-счетчика 3

Установка данного флага зависит от значений бит WGM3:0. В нормальном режиме и режиме CTC флаг TOV3 устанавливается при переполнении таймера-счетчика. См. табл. 61 для изучения поведения флага TOV3 при задании других значений WGM3:0. Флаг TOV3 автоматически сбрасывается при переходе на вектор прерывания по переполнению таймера-счетчика 3. Альтернативно флаг TOV3 сбрасывается путем записи в него лог. 1.

Разряд 1 – OCF3C: Флаг результата сравнения канала C таймера-счетчика 3

Данный флаг устанавливается следующим тактом после совпадения значения TCNT3 с регистром C порога сравнения (OCR3C).

Обратите внимание, что строб принудительной установки результата сравнения (FOC3C) не устанавливает флаг OCF3C. Флаг OCF3C автоматически сбрасывается при переходе на соответствующий вектор прерывания. Альтернативно, флаг OCF3C сбрасывается путем записи в него лог. 1.

Разряд 0 – OCF1C: Флаг результата сравнения канала C таймера-счетчика 1

Действует аналогично предыдущему, но в отношении TCNT1 и канала C таймера 1.

Предделители таймеров-счетчиков 1, 2 и 3

Таймеры-счетчики 1, 2 и 3 используют один и тот же модуль предделителя, но могут использовать различные установки предварительного деления. Приведенное ниже описание распространяется на все упомянутые таймеры.

Внутренний тактовый источник

Тактовый вход таймера-счетчика может быть непосредственно связан с системной синхронизацией, если установить CSn2:0 = 1. В данном случае достигается максимально быстрая работа таймера-счетчика на системной частоте fCLK_I/O. Альтернативно четыре производных тактовых сигнала на выходе предделителя могут использоваться в качестве тактового источника. Поделенный тактовый сигнал имеет частоту fCLK_I/O/8, fCLK_I/O/64, fCLK_I/O/256 или fCLK_I/O/1024.

Сброс предделителя

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 Tn, может использоваться как тактовый для таймеров-счетчиков (clkT1/clkT2/clkT3). Вывод Tn опрашивается каждый такт системной синхронизации логикой синхронизации данного вывода. Считанный таким образом сигнал проходит через детектор фронта. На рисунке 59 представлена функциональная схема синхронизации Tn и логики детектора фронта. Регистры тактируются положительным фронтом внутренней системной синхронизации (clkI/O). Детектор фронта генерирует один тактовый импульс clkT1/clkT2/clkT3 при определении положительного (CSn2:0 = 7) или отрицательного (CSn2:0 = 6) фронта.

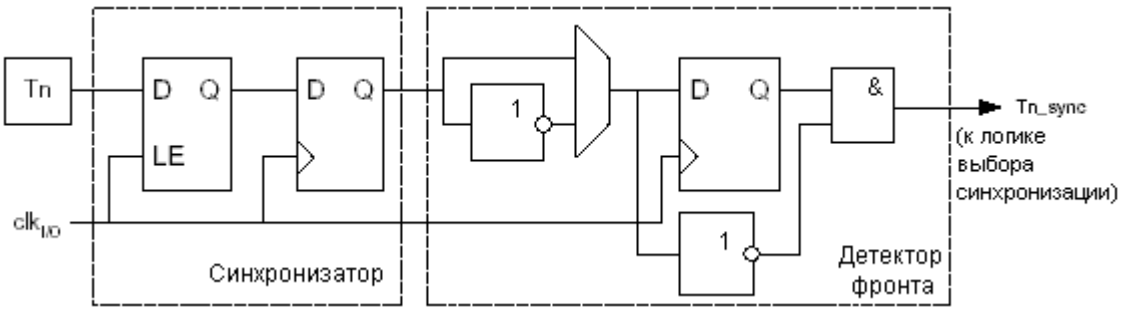


Рисунок 59 – Функциональная схема синхронизатора и детектора фронта вывода Tn

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

Разрешение и запрет тактового входа необходимо выполнять, когда Tn 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.

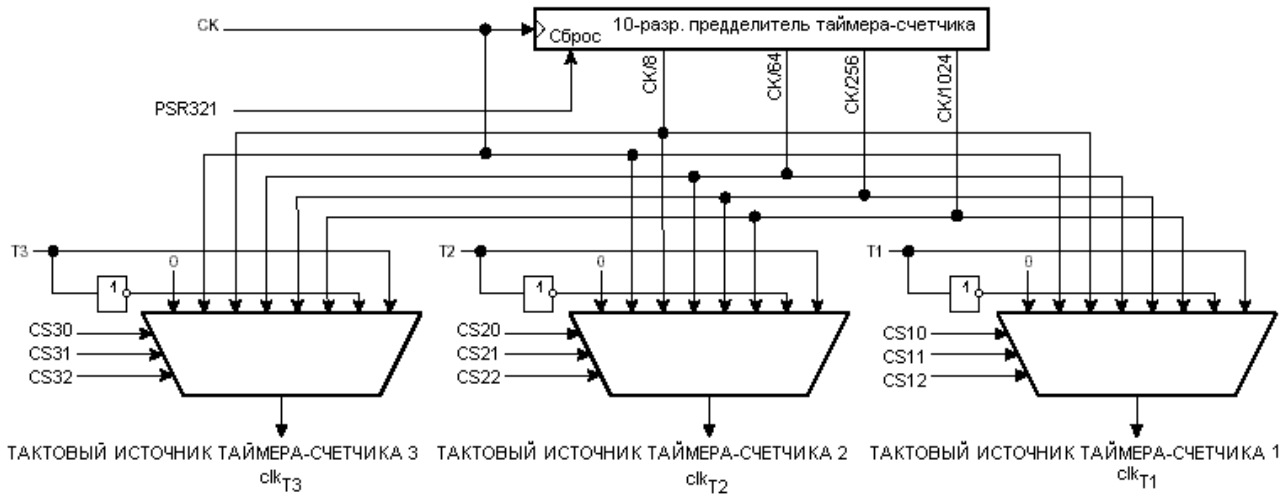


Рисунок 60 – Предделитель таймеров-счетчиков 1, 2 и 3

Прим.: логика синхронизации на входах T3/T2/T1 показана на рисунке 59.

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.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 7 – TSM: Режим синхронизации таймеров-счетчиков

Запись в данный бит лог. 1 активизирует режим синхронизации таймеров-счетчиков. В этом режиме запоминаются значения, записанные в биты PSR0 и PSR321, следовательно, запоминаются состояния соответствующих сигналов сброса предделителей. Этим гарантируется, что все соответствующие таймеры будут остановлены и им можно присвоить одинаковые значения без опасности их модификации в процессе конфигурации. Если в бит TSM записать лог. 0, то биты PSR0 и PSR321 сбросятся аппаратно и таймеры-счетчики начнут счет одновременно.

Разряд 0 – PSR321: Сброс предделителя таймеров-счетчиков 1, 2 и 3

Если данный бит равен лог. 1, то предделитель таймеров-счетчиков 1, 2 и 3 будет сброшен. Данный бит обычно сразу сбрасывается аппаратно за исключением, когда установлен бит TSM. Обратите внимание, что таймеры-счетчики 1, 2 и 3 используют один и тот же предделитель и сброс этого предделителя оказывает влияние на все три таймера.

Аналогово-цифровой преобразователь

Отличительные особенности:

10-разрядное разрешение
Интегральная нелинейность 0.5 мл. разр.
Абсолютная погрешность ±2 мл. разр.
Время преобразования 65 - 260 мкс.
Частота преобразования до 15 тыс. преобр. в сек. при максимальном разрешении
8 мультиплексированных однополярных входов
7 дифференциальных входных каналов
2 дифференциальных входных канала с опциональным усилением на 10 и 200
Представление результата с левосторонним или правосторонним выравниванием в 16-разр. слове

Диапазон входного напряжения АЦП 0…VCC
Выборочный внутренний ИОН на 2.56 В
Режимы одиночного преобразования и автоматического перезапуска
Прерывание по завершении преобразования АЦП
Механизм подавления шумов в режиме сна

ATmega128 содержит 10-разр. АЦП последовательного приближения. АЦП связан с 8-канальным аналоговым мультиплексором, 8 однополярных входов которого связаны с линиями порта F. Общий входных сигналов должен иметь потенциал 0В (т.е. связан с GND). АЦП также поддерживает ввод 16 дифференциальных напряжений. Два дифференциальных входа (ADC1, ADC0 и ADC3, ADC2) содержат каскад со ступенчатым программируемым усилением: 0 дБ (1x), 20 дБ (10x), или 46 дБ (200x). Семь дифференциальных аналоговых каналов используют общий инвертирующий вход (ADC1), а все остальные входы АЦП выполняют функцию неинвертирующих входов. Если выбрано усиление 1x или 10x, то можно ожидать 8-разр. разрешение, а если 200x, то 7-разрядное.

АЦП содержит УВХ (устройство выборки-хранения), которое поддерживает на постоянном уровне напряжение на входе АЦП во время преобразования. Функциональная схема АЦП показана на рисунке 108.

АЦП имеет отдельный вывод питания AVCC (аналоговое питание). AVCC не должен отличаться более чем на ± 0.3В от VCC. См. параграф “Подавитель шумов АЦП”, где приведены рекомендации по подключению этого вывода.

В качестве внутреннего опорного напряжения может выступать напряжение от внутреннего ИОНа на 2.56В или напряжение AVCC. Если требуется использование внешнего ИОН, то он должен быть подключен к выводу AREF с подключением к этому выводу блокировочного конденсатора для улучшения шумовых характеристик.

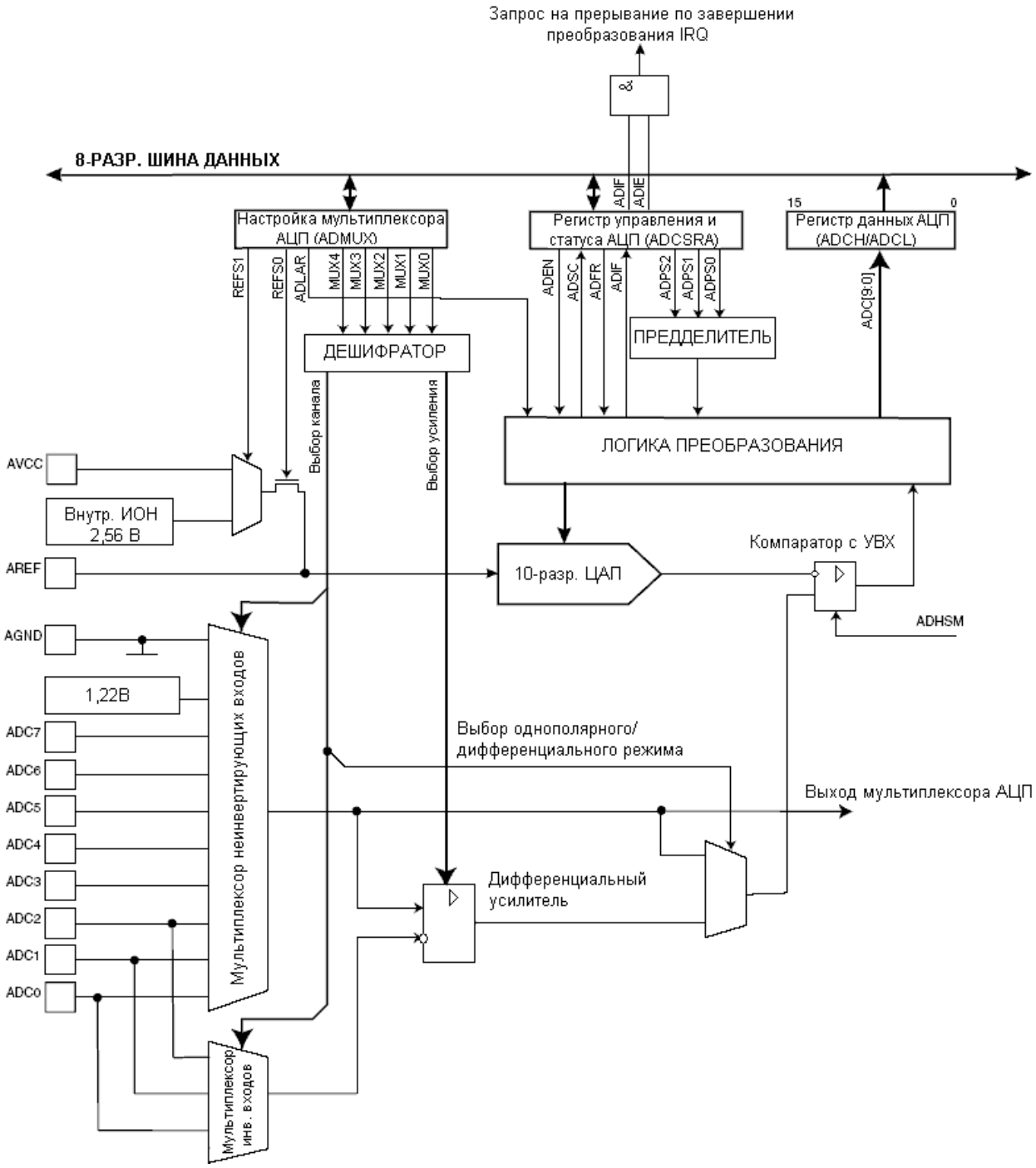


Рисунок 108- Функциональная схема аналогово-цифрового преобразователя

Принцип действия

АЦП преобразовывает входное аналоговое напряжение в 10-разр. код методом последовательных приближений. Минимальное значение соответствует уровню GND, а максимальное уровню AREF минус 1 мл. разр. К выводу AREF опционально может быть подключено напряжение AVCC или внутренний ИОН на 1.22В путем записи соответствующих значений в биты REFSn в регистр ADMUX. Несмотря на то, что ИОН на 2.56В находится внутри микроконтроллера, к его выходу может быть подключен блокировочный конденсатор для снижения чувствительности к шумам, т.к. он связан с выводом AREF.

Канал аналогового ввода и каскад дифференциального усиления выбираются путем записи бит MUX в регистр ADMUX. В качестве однополярного аналогового входа АЦП может быть выбран один из входов ADC0…ADC7, а также GND и выход фиксированного источника опорного

напряжения 1,22 В. В режиме дифференциального ввода предусмотрена возможность выбора инвертирующих и неинвертирующих входов к дифференциальному усилителю.

Если выбран дифференциальный режим аналогового ввода, то дифференциальный усилитель будет усиливать разность напряжений между выбранной парой входов на заданный коэффициент усиления. Усиленное таким образом значение поступает на аналоговый вход АЦП. Если выбирается однополярный режим аналогового ввода, то каскад усиления пропускается

Работа АЦП разрешается путем установки бита ADEN в ADCSRA. Выбор опорного источника и канала преобразования не возможно выполнить до установки ADEN. Если ADEN = 0, то АЦП не потребляет ток, поэтому, при переводе в экономичные режимы сна рекомендуется предварительно отключить АЦП.

АЦП генерирует 10-разрядный результат, который помещается в пару регистров данных АЦП ADCH и ADCL. По умолчанию результат преобразования размещается в младших 10-ти разрядах 16-разр. слова (выравнивание справа), но может быть опционально размещен в старших 10-ти разрядах (выравнивание слева) путем установки бита ADLAR в регистре ADMUX.

Практическая полезность представления результата с выравниванием слева существует, когда достаточно 8-разрядное разрешение, т.к. в этом случае необходимо считать только регистр ADCH. В другом же случае необходимо первым считать содержимое регистра ADCL, а затем ADCH, чем гарантируется, что оба байта являются результатом одного и того же преобразования. Как только выполнено чтение ADCL блокируется доступ к регистрам данных со стороны АЦП. Это означает, что если считан ADCL и преобразование завершается перед чтением регистра ADCH, то ни один из регистров не может модифицироваться и результат преобразования теряется. После чтения ADCH доступ к регистрам ADCH и ADCL со стороны АЦП снова разрешается.

АЦП генерирует собственный запрос на прерывание по завершении преобразования. Если между чтением регистров ADCH и ADCL запрещен доступ к данным для АЦП, то прерывание возникнет, даже если результат преобразования будет потерян.

Запуск преобразования

Одиночное преобразование запускается путем записи лог. 1 в бит запуска преобразования АЦП ADSC. Данный бит остается в высоком состоянии в процессе преобразования и сбрасывается по завершении преобразования. Если в процессе преобразования переключается канал аналогового ввода, то АЦП автоматически завершит текущее преобразование прежде, чем переключит канал.

В режиме автоматического перезапуска АЦП непрерывно оцифровывает аналоговый сигнал и обновляет регистр данных АЦП. Данный режим задается путем записи лог. 1 в бит ADFR регистра ADCSRA. Первое преобразование инициируется путем записи лог. 1 в бит ADSC регистра ADCSRA. В данном режиме АЦП выполняет последовательные преобразования, независимо от того сбрасывается флаг прерывания АЦП ADIF или нет.

Предделитель и временная диаграмма преобразования

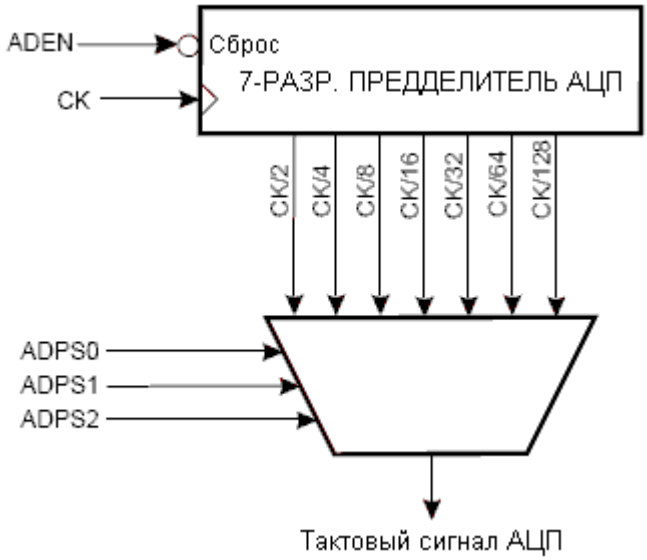


Рисунок 109 – Предделитель АЦП

Если требуется максимальная разрешающая способность (10 разрядов), то частота на входе схемы последовательного приближения должна быть в диапазоне 50…200 кГц. Если достаточно разрешение менее 10 разрядов, но требуется более высокая частота преобразования, то частота на входе АЦП может быть установлена свыше 200 кГц.

Модуль АЦП содержит предделитель, который формирует производные частоты свыше 100 кГц по отношению к частоте синхронизации ЦПУ. Коэффициент деления устанавливается с помощью бит ADPS в регистре ADCSRA. Предделитель начинает счет с момента включения АЦП установкой бита ADEN в регистре ADCSRA. Предделитель работает пока бит ADEN = 1 и сброшен, когда ADEN=0.

Если инициируется однополярное преобразование установкой бита ADSC в регистре ADCSRA, то преобразование начинается со следующего нарастающего фронта тактового сигнала АЦП. Особенности временной диаграммы дифференциального преобразования представлены в “Каналы дифференциального усиления”.

Нормальное преобразование требует 13 тактов синхронизации АЦП. Первое преобразование после включения АЦП (установка ADEN в ADCSRA) требует 25 тактов синхронизации АЦП за счет необходимости инициализации аналоговой схемы.

После начала нормального преобразования на выборку-хранение затрачивается 1.5 такта синхронизации АЦП, а после начала первого преобразования – 13,5 тактов. По завершении преобразования результат помещается в регистры данных АЦП и устанавливается флаг ADIF. В режиме одиночного преобразования одновременно сбрасывается бит ADSC. Программно бит ADSC может быть снова установлен и новое преобразование будет инициировано первым нарастающим фронтом тактового сигнала АЦП.

В режиме автоматического перезапуска новое преобразование начинается сразу по завершении предыдущего, при этом ADSC остается в высоком состоянии. Времена преобразования для различных режимов преобразования представлены в таблице 95.

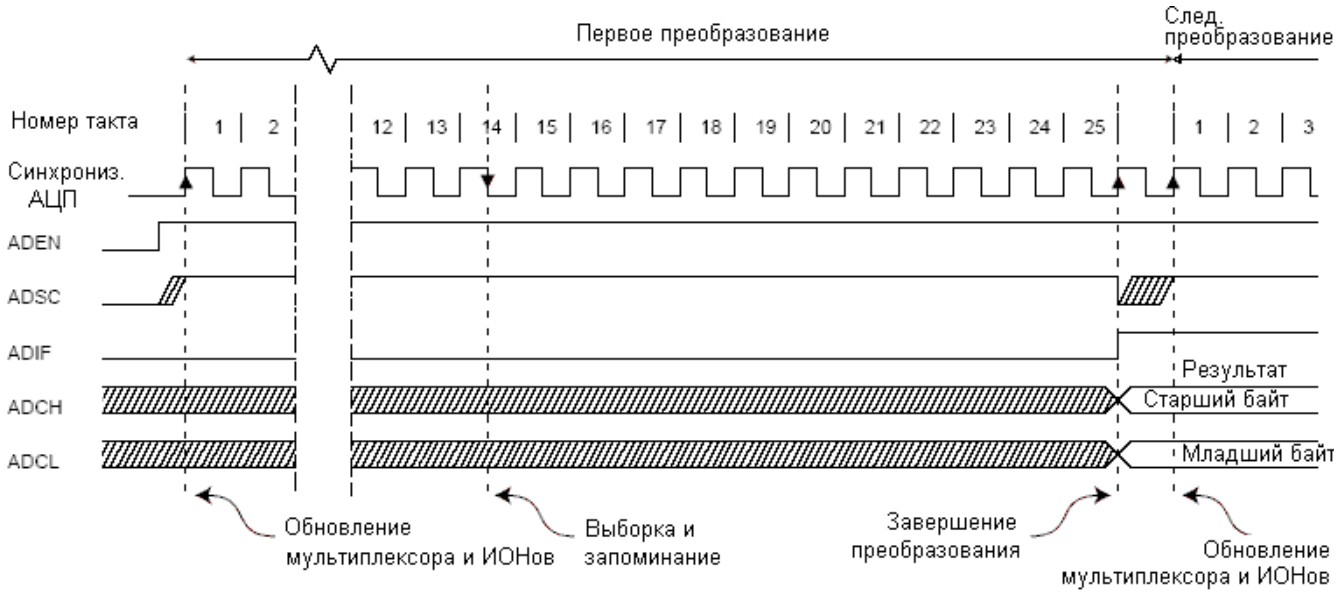


Рисунок 110 – Временная диаграмма работы АЦП при первом преобразовании в режиме одиночного преобразования

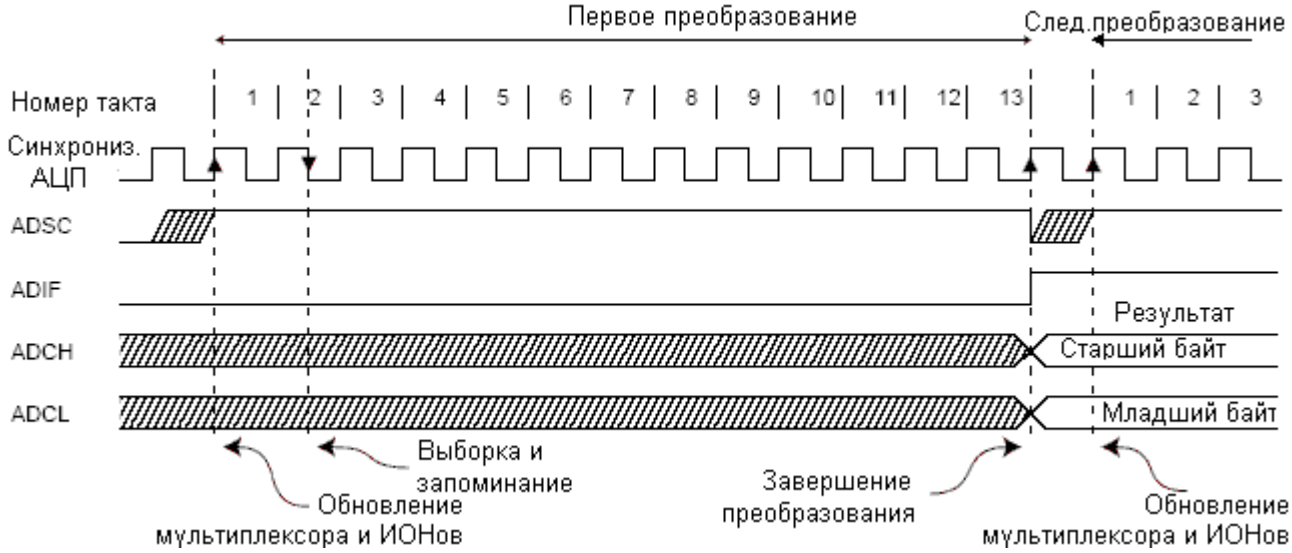


Рисунок 111 – Временная диаграмма работы АЦП в режиме одиночного преобразования

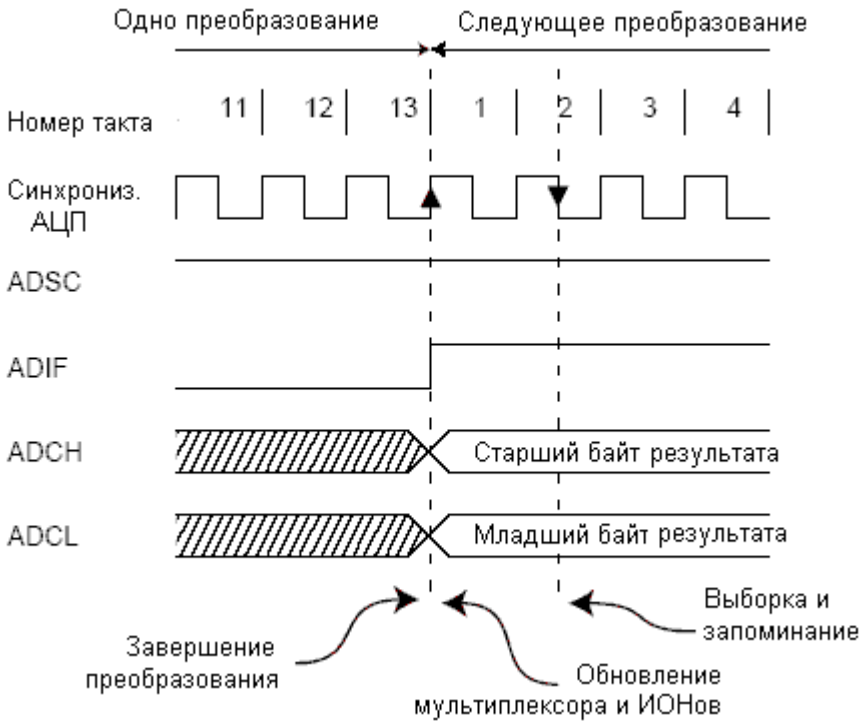


Рисунок 112 – Временная диаграмма работы АЦП в режиме автоматического перезапуска

Таблица 95 – 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
Нормальное дифференциальное преобразование	1.5/2.5	13/14

Каналы дифференциального усиления

Если используются каналы дифференциального усиления, то необходимо принять во внимание некоторые особенности.

Дифференциальные преобразования синхронизированы по отношению к внутренней синхронизации СКАЦП2, частого которого равна половине частоты синхронизации АЦП. Данная синхронизация выполняется автоматически интерфейсом АЦП таким образом, чтобы выборка-хранение инициировалась определенным фронтом СКАЦП2. Если преобразование (все одиночные преобразования и первое преобразование в режиме автоматического перезапуска) инициировалось пользователем, когда СКАЦП2 находился в низком лог. состоянии, то его длительность будет эквивалента однополярному преобразованию (13 тактов синхронизации АЦП). Если преобразование инициируется пользователем, когда СКАЦП2 равен лог. 1 , оно будет длиться 14 тактов синхронизации АЦП вследствие работы механизма синхронизации. В режиме автоматического перезапуска новое преобразование инициируется сразу по завершении предыдущего, а т.к. в этот момент СКАЦП2 равен лог. 1, то все преобразования, которые были автоматически перезапущены (т.е. все, кроме первого), будут длиться 14 тактов синхронизации АЦП. Усилительный каскад оптимизирован под частотный диапазон до 4 кГц для любых коэффициентов усиления. Усиление сигналов более высоких частот будет нелинейным. Поэтому, если входной сигнал содержит частотные составляющие выше частотного диапазона усилительного каскада, то необходимо установить внешний фильтр низких частот. Обратите внимание, что частота синхронизации АЦП не связана с ограничением по частотному диапазону

усилительного каскада. Например, период синхронизации АЦП может быть 6 мкс, при котором частота преобразования канала равна 12 тыс. преобр. в секунду, независимо от частотного диапазона этого канала.

Изменение канала или выбор опорного источника

Биты MUXn и REFS1:0 в регистре ADMUX поддерживают одноступенчатую буферизацию через временный регистр. Этим гарантируется, что новые настройки канала преобразования и опорного источника вступят в силу в безопасный момент для преобразования. До начала преобразования любые изменения канала и опорного источника вступаю в силу сразу после их модификации. Как только начинается процесс преобразования доступ к изменению канала и опорного источника блокируется, чем гарантируется достаточность времени на преобразование для АЦП. Непрерывность модификации возвращается на последнем такте АЦП перед завершением преобразования (перед установкой флага ADIF в регистре ADCSRA). Обратите внимание, что преобразование начинается следующим нарастающим фронтом тактового сигнала АЦП после записи ADSC. Таким образом, пользователю не рекомендуется записывать новое значение канала или опорного источника в ADMUX до 1-го такта синхронизации АЦП после записи ADSC.

Особые меры необходимо предпринять при изменении дифференциального канала. Как только осуществлен выбор дифференциального канала усилительному каскаду требуется 125 мкс для стабилизации нового значения. Следовательно, в течение первых после переключения дифференциального канала 125 мкс не должно стартовать преобразование. Если же в этот период преобразования все-таки выполнялись, то их результат необходимо игнорировать.

Такую же задержку на установление необходимо ввести при первом дифференциальном преобразовании после изменения опорного источника АЦП (за счет изменения бит REFS1:0 в ADMUX).

Если разрешена работа интерфейса JTAG, то функции каналов АЦП на выводах порта F 7…4 отменяется. См. табл. 42 и “Альтернативные функции порта F”.

Входные каналы АЦП

При переключении входного канала необходимо учесть некоторые рекомендации, которые исключат некорректность переключения.

В режиме одиночного преобразования переключение канала необходимо выполнять перед началом преобразования. Переключение канала может произойти только в течение одного такта синхронизации АЦП после записи лог. 1 в ADSC. Однако самым простым методом является ожидание завершения преобразования перед выбором нового канала.

В режиме автоматического перезапуска канал необходимо выбирать перед началом первого преобразования. Переключение канала происходит аналогично - в течение одного такта синхронизации АЦП после записи лог. 1 в ADSC. Но самым простым методом является ожидание завершения перового преобразования, а затем переключение канала. Поскольку следующее преобразование уже запущено автоматически, то следующий результат будет соответствовать предыдущему каналу. Последующие преобразования отражают результат для нового канала.

При переключении на дифференциальный канал первое преобразование будет характеризоваться плохой точностью из-за переходного процесса в схеме автоматической регулировки смещения. Следовательно, первый результат такого преобразования рекомендуется игнорировать.

Источник опорного напряжения АЦП

Источник опорного напряжения (ИОН) для АЦП (VИОН) определяет диапазон преобразования АЦП. Если уровень однополярного сигнала свыше VИОН, то результатом преобразования будет 0x3FF. В качестве VИОН могут выступать AVCC, внутренний ИОН 2,56В или внешний ИОН, подключенный к выв. AREF. AVCC подключается к АЦП через пассивный ключ. Внутреннее опорное напряжение 2,56В генерируется внутренним эталонным источником VBG,

буферизованного внутренним усилителем. В любом случае внешний вывод AREF связан непосредственно с АЦП и, поэтому, можно снизить влияние шумов на опорный источник за счет подключения конденсатора между выводом AREF и общим. Напряжение VИОН также может быть измерено на выводе AREF высокоомным вольтметром. Обратите внимание, что VИОН является высокоомным источником и, поэтому, внешне к нему может быть подключена только емкостная нагрузка.

Если пользователь использует внешний опорный источник, подключенный к выв. AREF, то не допускается использование другой опции опорного источника, т.к. это приведет к шунтированию внешнего опорного напряжения. Если к выв. AREF не приложено напряжение, то пользователь может выбрать AVCC и 2.56В качестве опорного источника. Результат первого преобразования после переключения опорного источника может характеризоваться плохой точностью и пользователю рекомендуется его игнорировать.

Если используются дифференциальные каналы, то выбранный опорный источник должен быть меньше уровня AVCC, что показано в табл. 136.

Подавитель шумов АЦП

АЦП характеризуется возможностью подавления шумов, которые вызваны работой ядра ЦПУ и периферийных устройств ввода-вывода. Подавитель шумов может быть использован в режиме снижения шумов АЦП и в режиме холостого хода. При использовании данной функции необходимо придерживаться следующей процедуры:

1. Убедитесь, что работа АЦП разрешена и он не выполняет преобразования. Выберите режим одиночного преобразования и разрешите прерывание по завершении преобразования.
2. Введите режим уменьшения шумов АЦП (или режим холостого хода). АЦП запустит преобразование как только остановится ЦПУ.
3. Если до завершения преобразования не возникает других прерываний, то АЦП вызовет прерывание ЦПУ и программа перейдет на вектор обработки прерывания по завершении преобразования АЦП. Если до завершения преобразования другое прерывание пробуждает микроконтроллер, то это прерывание обрабатывается, а по завершении преобразования генерируется соответствующий запрос на прерывание. АЦП остается в активном режиме пока не будет выполнена очередная команда sleep.

Обратите внимание, что АЦП не отключается автоматически при переводе во все режимы сна, кроме режима холостого хода и снижения шумов АЦП. Поэтому, пользователь должен предусмотреть запись лог. 0 в бит ADEN перед переводом в такие режимы сна во избежание чрезмерного энергопотребления. Если работа АЦП была разрешена в таких режимах сна и пользователь желает выполнить дифференциальное преобразование, то после пробуждения необходимо включить, а затем выключить АЦП для инициации расширенного преобразования, чем будет гарантировано получение действительного результата.

Схема аналогового входа

Схема аналогового входа для однополярных каналов представлена на рисунке 113. Независимо от того, какой канал подключен к АЦП, аналоговый сигнал, подключенный к выв. ADCn, нагружается емкостью вывода и входным сопротивлением утечки. После подключения канала к АЦП аналоговый сигнал будет связан с конденсатором выборки-хранения через последовательный резистор, сопротивление которого эквивалентно всей входной цепи.

АЦП оптимизирован под аналоговые сигналы с выходным сопротивлением не более 10 кОм. Если используется такой источник сигнала, то время выборки незначительно. Если же используется источник с более высоким входным сопротивлением, то время выборки будет определяться временем, которое требуется для зарядки конденсатора выборки-хранения источником аналогового сигнала. Рекомендуется использовать источники только с малым выходным сопротивлением и медленно изменяющимися сигналами, т.к. в этом случае будет достаточно быстрым заряд конденсатора выборки-хранения.

По отношению к каналам с дифференциальным усилением рекомендуется использовать сигналы с внутренним сопротивлением до нескольких сотен кОм. Следует предусмотреть, чтобы в предварительных каскадах формирования аналогового сигнала ко входу АЦП не вносились частоты выше fАЦП/2, в противном случае результат преобразования может быть некорректным. Если вероятность проникновения высоких частот существует, то рекомендуется перед АЦП установить фильтр низких частот.

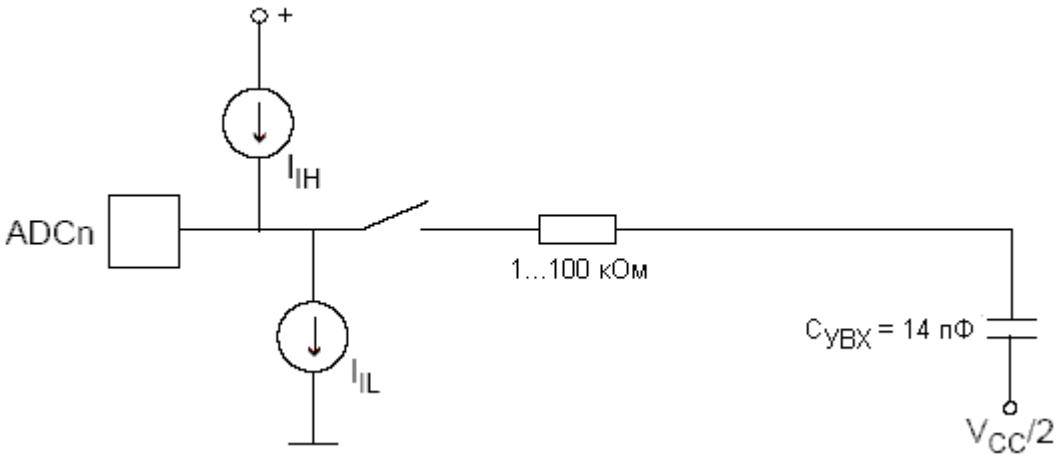


Рисунок 113 – Схема аналогового входа

Рекомендации по снижению влияния шумов на результат преобразования

Работа цифровых узлов внутри и снаружи микроконтроллера связана с генерацией электромагнитных излучений, которые могут негативно сказаться на точность измерения аналогового сигнала. Если точность преобразования является критическим параметром, то уровень шумов можно снизить, придерживаясь следующих рекомендаций:

1. Выполняйте путь аналоговых сигналов как можно более коротким. Следите, чтобы аналоговые сигналы проходили над плоскостью (слоем) с аналоговой землей (экраном) и далеко от проводников, передающих высокочастотные цифровые сигналы.
2. Вывод AVCC необходимо связать с цифровым питанием VCC через LC-цепь в соответствии с рис. 114.
3. Используйте функцию подавления шумов АЦП, внесенных работой ядра ЦПУ.
4. Если какой-либо из выводов АЦП используется как цифровой выход, то чрезвычайно важно не допустить переключение состояния этого выхода в процессе преобразования.

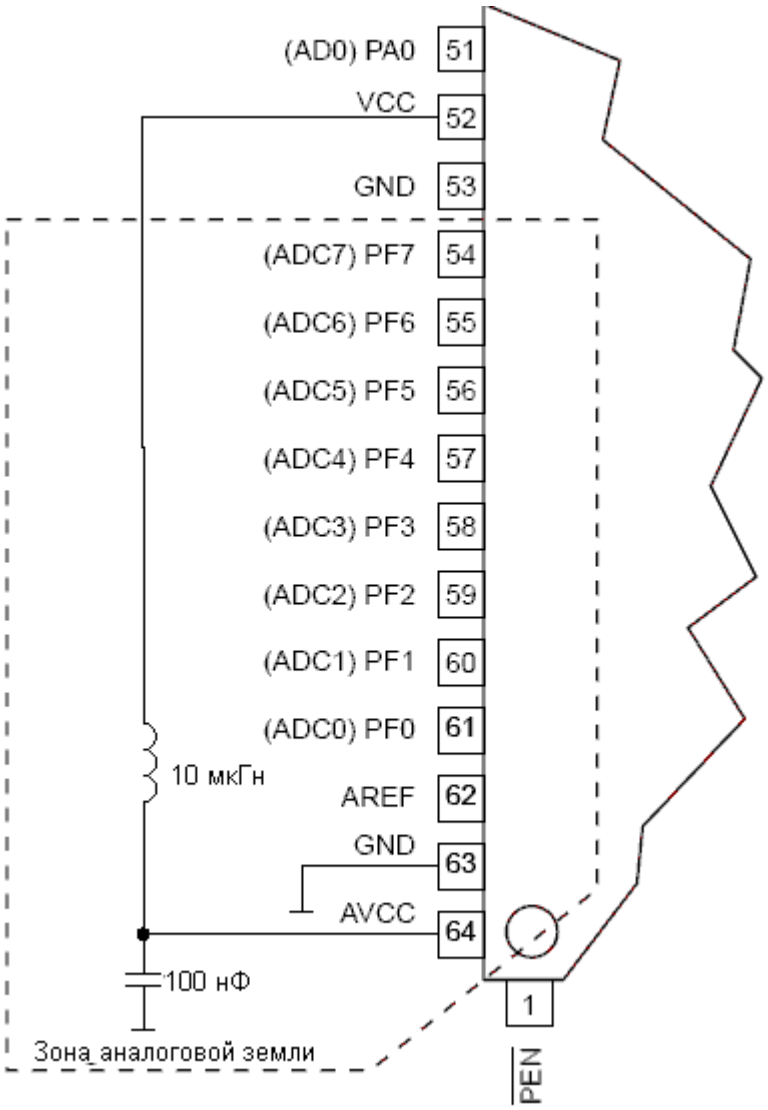


Рисунок 114 – Подключение питания АЦП

Методы компенсации смещения

Усилительный каскад имеет встроенную схему компенсации смещения, которая стремится максимально приблизить к нулю смещение дифференциального измерения. Оставшееся смещение можно измерить, если в качестве дифференциальных входов АЦП выбрать один и тот же вывод микроконтроллера. Измеренное таким образом остаточное смещение можно программно вычесть из результата преобразования. Использование программного алгоритма коррекции смещения позволяет уменьшить смещение ниже одного мл. разр.

Определения погрешностей аналогово-цифрового преобразования

n-разрядный однополярный АЦП преобразовывает напряжение линейно между GND и VИОН с количеством шагами 2n (мл. разрядов). Минимальный код = 0, максимальный = 2n-1. Основные погрешности преобразования являются отклонением реальной функции преобразования от идеальной. К ним относятся:

Смещение – отклонение первого перехода (с 0x000 на 0x001) по сравнению с идеальным переходом (т.е. при 0.5 мл. разр.). Идеальное значение : 0 мл. разр.

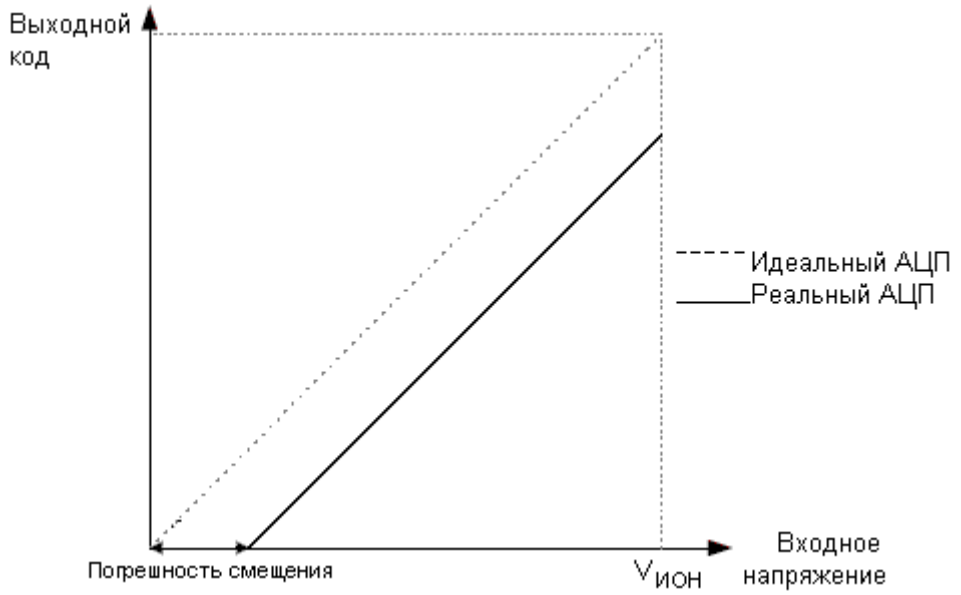


Рисунок 115 – Погрешность смещения

Погрешность усиления. После корректировки смещения погрешность усиления представляет собой отклонение последнего перехода (с 0x3FE на 0x3FF) от идеального перехода (т.е. отклонение при максимальном значении минус 1,5 мл. разр.). Идеальное значение: 0 мл. разр.

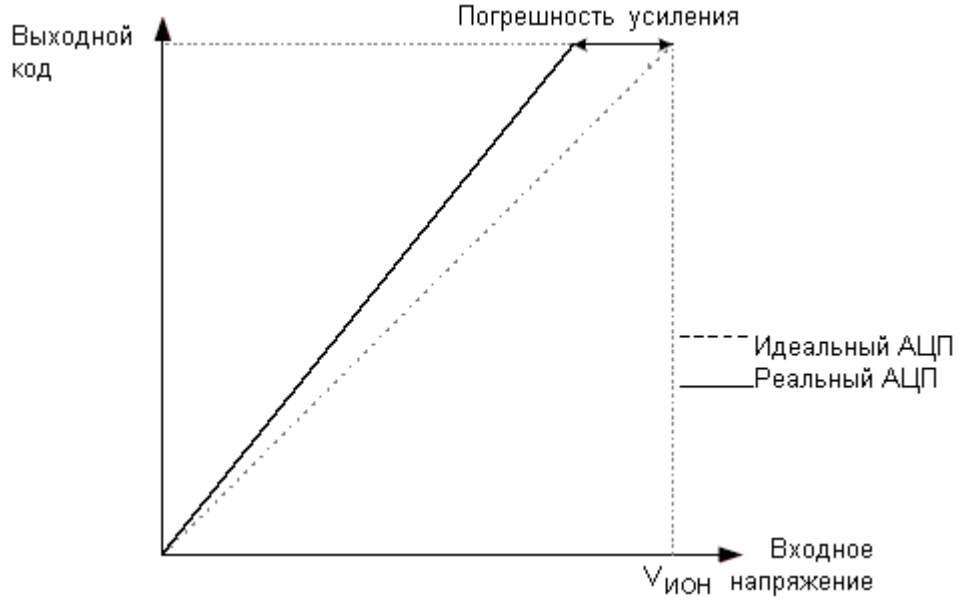


Рисунок 116 – Погрешность усиления

Интегральная нелинейность (ИНЛ). После корректировки смещения и погрешности усиления ИНЛ представляет собой максимальное отклонение реальной функции преобразования от идеальной для любого кода. Идеальное значение ИНЛ = 0 мл. разр.

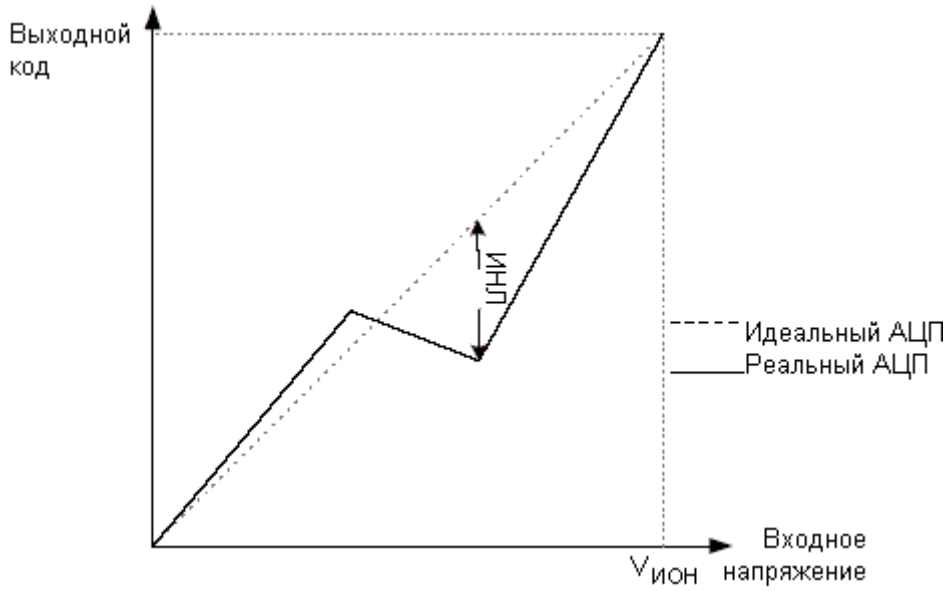


Рисунок 117- Интегральная нелинейность (ИНЛ)

Дифференциальная нелинейность (ДНЛ). Максимальное отклонение между шириной фактического кода (интервал между двумя смежными переходами) от ширины идеального кода (1 мл. разр.). Идеальное значение: 0 мл. разр.

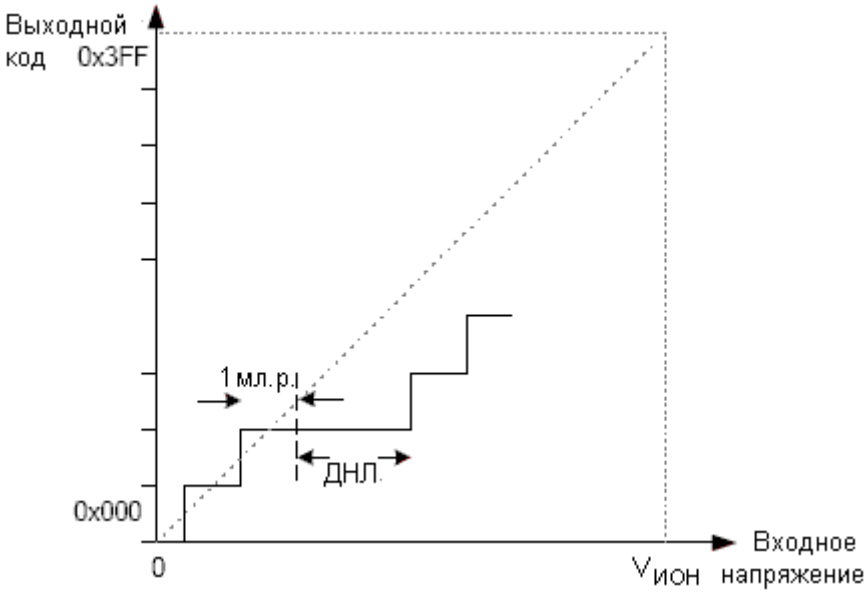


Рисунок 118- Дифференциальная нелинейность (ДНЛ)

Погрешность квантования. Возникает из-за преобразования входного напряжения в конечное число кодов. Погрешность квантования- интервал входного напряжения протяженностью 1 мл. разр. (шаг квантования по напряжению), который характеризуется одним и тем же кодом. Всегда равен ±0.5 мл. разр.

Абсолютная погрешность. Максимальное отклонение реальной (без подстройки) функции преобразования от реальной при любом коде. Является результатом действия нескольких эффектов: смещение, погрешность усиления, дифференциальная погрешность, нелинейность и погрешность квантования. Идеальное значение: ±0.5 мл. разр.

Результат преобразования АЦП

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

Для однополярного преобразования:

где Vвх – уровень напряжения на подключенном к АЦП входу;

Vион –напряжение выбранного источника опорного напряжения (см. табл. 97 и табл. 98). Код 0x000 соответствует уровню аналоговой земли, а 0x3FF - уровню напряжения ИОН минус 1 шаг квантования по напряжению. При использовании дифференциального канала

Результат представляется в коде двоичного дополнения, начиная с 0x200 (-512d) до 0x1FF (+511d). Обратите внимание, что при необходимости быстро определить полярность результата достаточно опросить старший бит результата преобразования (ADC9 в ADCH). Если данный бит равен лог. 1, то результат отрицательный, если же лог. 0, то положительный. На рисунке 119 представлена функция преобразования АЦП в дифференциальном режиме.

В таблице 96 представлены результирующие выходные коды для дифференциальной пары каналов (ADCn - ADCm) с коэффициентом усиления Ку и опорным напряжением VИОН.

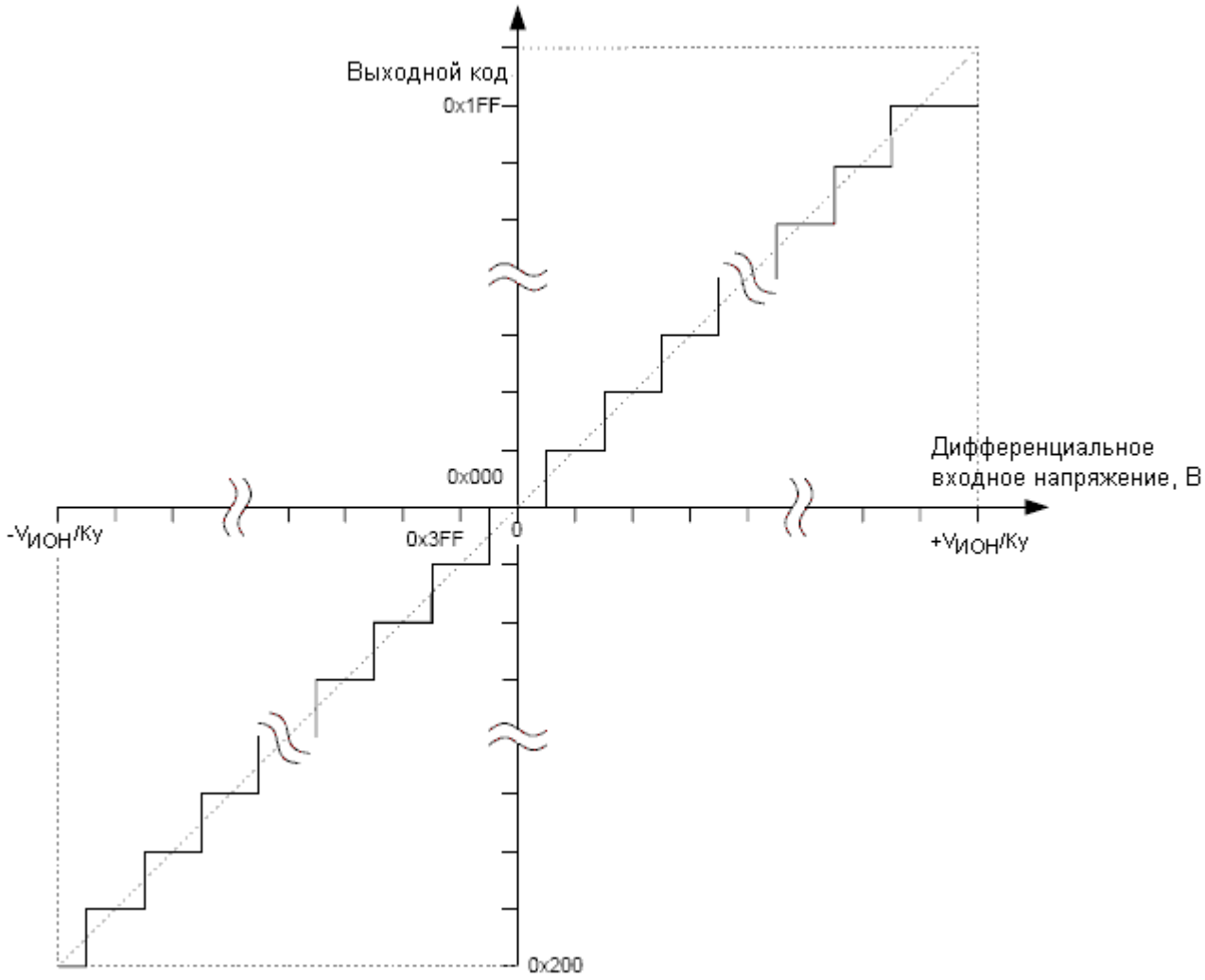


Рисунок 119 – Функция преобразования АЦП при измерении дифференциального сигнала

Таблица 96 – Связь между входным напряжением и выходными кодами

VАЦПn	**Считываемый код**	**Соответствующее десятичное значение**
VАЦПm + VИОН /Ку	0x1FF	511
VАЦПm + 0.999 VИОН / Ку	0x1FF	511
VАЦПn + 0.998 VИОН / Ку	0x1FE	510

…	...	...
VАЦПm + 0.001 VИОН / Ку	0x001	1
VАЦПm	0x000	0
VАЦПm - 0.001 VИОН / Ку	0x3FF	-1
…	...	...
VАЦПm - 0.999 VИОН / Ку	0x201	-511
VАЦПm – VИОН / Ку	0x200	-512

Пример: Пусть ADMUX = 0xED (пара входов ADC3 - ADC2, Ку=1, Vион=2.56В, результат с левосторонним выравниванием), напряжение на входе ADC3 = 300 мВ, а на входе ADC2 = 500 мВ, тогда:

КодАЦП = 512 * 10 * (300 - 500) / 2560 = -400 = 0x270

С учетом выбранного формата размещения результата (левосторонний) ADCL = 0x00, а ADCH = 0x9C. Если же выбран правосторонний формат (ADLAR=0), то ADCL = 0x70, ADCH = 0x02.

Регистр управления мультиплексором АЦП– ADMUX

Разряд	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	**ADMUX**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 7:6 – REFS1:0: Биты выбора источника опорного напряжения

Данные биты определяют какое напряжение будет использоваться в качестве опорного для АЦП (см. табл. 97). Если изменить значения данных бит в процессе преобразования, то новые установки вступят в силу только по завершении текущего преобразования (т.е. когда установится бит ADIF в регистре ADCSRA). Внутренний ИОН можно не использовать, если к выводу AREF подключен внешний опорный источник.

Таблица 97 – Выбор опорного источника АЦП

REFS1	REFS0	**Опорный источник**
0	0	AREF, внутренний VИОН 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

Разряд 5 – ADLAR: Бит управления представлением результата преобразования

Бит ADLAR влияет на представление результата преобразования в паре регистров результата преобразования АЦП. Если ADLAR = 1, то результат преобразования будет иметь левосторонний формат, в противном случае - правосторонний. Действие бита ADLAR вступает в силу сразу после изменения, независимо от выполняющегося параллельно преобразования. Полное описание действия данного бита представлено в “Регистры данных АЦП – ADCL и ADCH”.

Разряд 4:0 – MUX4:0: Биты выбора аналогового канала и коэффициента усиления

Данные биты определяют какие из имеющихся аналоговых входов подключаются к АЦП. Кроме того, с их помощью можно выбрать коэффициент усиления для дифференциальных каналов (см.

табл. 98). Если значения бит изменить в процессе преобразования, то механизм их действия вступит в силу только после завершения текущего преобразования (после установки бита ADIF в регистре ADCSRA).

Таблица 98 – Выбор входного канала и коэффициента усиления

MUX4..0	Однополярный вход	Неинвертирующий дифференциальный вход	Инвертирующий дифференциальный вход	Коэффициент усиления, Ку
00000	ADC0	Нет		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000	Нет	ADC0	ADC0	10
01001		ADC1	ADC0	10
01010		ADC0	ADC0	200
01011		ADC1	ADC0	200
01100		ADC2	ADC2	10
01101		ADC3	ADC2	10
01110		ADC2	ADC2	200
01111		ADC3	ADC2	200
10000		ADC0	ADC1	1
10001		ADC1	ADC1	1
10010		ADC2	ADC1	1
10011		ADC3	ADC1	1
10100		ADC4	ADC1	1
10101		ADC5	ADC1	1
10110		ADC6	ADC1	1
10111		ADC7	ADC1	1
11000		ADC0	ADC2	1
11001		ADC1	ADC2	1
11010		ADC2	ADC2	1
11011		ADC3	ADC2	1
11100		ADC4	ADC2	1
11101		ADC5ADC21		
11110	1.23В (VBG)	Нет111110В(GND)		

Регистр А управления и статуса АЦП – ADCSRA

Разряд	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	**ADCSRA**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 7 – ADEN: Разрешение работы АЦП

Запись в данный бит лог. 1 разрешает работу АЦП. Если в данный бит записать лог. 0, то АЦП отключается, даже если он находился в процессе преобразования.

Разряд 6 – ADSC: Запуск преобразования АЦП

В режиме одиночного преобразования установка данного бита инициирует старт каждого преобразования. В режиме автоматического перезапуска установкой этого бита инициируется только первое преобразование, а все остальные выполняются автоматически. Первое преобразование после разрешения работы АЦП, инициированное битом ADSC, выполняется по расширенному алгоритму и длится 25 тактов синхронизации АЦП, вместо обычных 13 тактов. Это связано с необходимостью инициализации АЦП.

В процессе преобразования при опросе бита ADSC возвращается лог. 1, а по завершении преобразования – лог. 0. Запись лог. 0 в данный бит не предусмотрено и не оказывает никакого действия.

Разряд 5 – ADFR: Выбор режима автоматического перезапуска АЦП

Если в данный бит записать лог. 1, то АЦП перейдет в режим автоматического перезапуска. В этом режиме АЦП автоматически выполняет преобразования и модифицирует регистры результата преобразования через фиксированные промежутки времени. Запись лог. 0 в этот бит прекращает работу в данном режиме.

Разряд 4 – ADIF: Флаг прерывания АЦП

Данный флаг устанавливается после завершения преобразования АЦП и обновления регистров данных. Если установлены биты ADIE и I (регистр SREG), то происходит прерывание по завершении преобразования. Флаг ADIF сбрасывается аппаратно при переходе на соответствующий вектор прерывания. Альтернативно флаг ADIF сбрасывается путем записи лог. 1 в него. Обратите внимание, что при выполнении команды "чтение-модификация-запись" с регистром ADCSRA ожидаемое прерывание может быть отключено. Данное также распространяется на использование инструкций SBI и CBI.

Разряд 3 – ADIE: Разрешение прерывания АЦП

После записи лог. 1 в этот бит, при условии, что установлен бит I в регистре SREG, разрешается прерывание по завершении преобразования АЦП.

Разряды 2:0 – ADPS2:0: Биты управления предделителем АЦП

Данные биты определяют на какое значение тактовая частота ЦПУ будет отличаться от частоты входной синхронизации АЦП.

Таблица 99 – Управление предделителем АЦП

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

Регистры данных АЦП – ADCL и ADCH

ADLAR = 0:

Разряд	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	
Чтение/запись	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	
	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	
Исх. значение	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1:

Разряд	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	
Чтение/запись	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	
	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	
Исх. значение	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

По завершении преобразования результат помещается в этих двух регистрах. При использовании дифференциального режима преобразования результат представляется в коде двоичного дополнения.

Если выполнено чтение ADCL, то доступ к этим регистрам для АЦП будет заблокирован (т.е. АЦП не сможет в дальнейшем модифицировать результат преобразования), пока не будет считан регистр ADCH.

Левосторонний формат представления результата удобно использовать, если достаточно 8 разрядов. В этом случае 8-разрядный результат хранится в регистре ADCH и, следовательно, чтение регистра ADCL можно не выполнять. При правостороннем формате необходимо сначала считать ADCL, а затем ADCH.

ADC9:0: Результат преобразования АЦП

Данные биты представляют результат преобразования.

Интерфейс JTAG и встроенная отладочная система

Отличительные особенности:

Интерфейс JTAG (совместимость со стандартом IEEE 1149.1)
Функции граничного сканирования в соответствии с IEEE 1149.1 (JTAG)
Отладчик имеет доступ к следующим блокам микроконтроллера:
– Все внутренние периферийные блоки
– Внутреннее и внешнее ОЗУ
– Внутренний файл регистров
– Программный счетчик
– ЭСППЗУ и флэш-память

Отладочная система поддерживает обширные условия прерывания, в т.ч.:
– Прерывания по инструкциям AVR-микроконтроллера
– Прерывание по изменению потока памяти программ
– Пошаговое прерывание
– Точки прерывания памяти программ по одиночному адресу или адресному диапазону
– Точки прерывания памяти данных по одиночному адресу или адресному диапазону
Программирование флэш-памяти, ЭСППЗУ, конфигурационных бит и бит защиты программы через интерфейс JTAG
Встроенная отладочная система поддерживается AVR Studio

Введение

Интерфейс JTAG микроконтроллеров семейства AVR совместим со стандартом IEEE 1149.1 и может использоваться в следующих целях:

Тестирование печатных плат с помощью функции граничного сканирования
Программирование энергонезависимой памяти, конфигурационных бит и бит защиты программы
Встроенная отладка

Встроенная отладочная система управляется через специальные JTAG-инструкции, которые известны только внутри корпорации ATMEL и выбранным ATMEL сторонним поставщикам отладочных средств.

На рисунке 120 представлена функциональная схема интерфейса JTAG и встроенной отладочной системы. TAP-контроллер – цифровой автомат, который управляется сигналами TCK и TMS. TAP-контроллер выбирает в качестве сканируемой цепи (сдвигового регистра) между входом TDI и выходом TDO или регистр JTAG-инструкции или один из нескольких регистров данных. В регистре инструкции сохраняются JTAG-инструкции, которые управляют поведением регистра данных.

Идентификационный (ID) регистр, регистр пропуска и регистры цепи граничного сканирования и данных используются для тестирования на уровне проверки печатной платы. Интерфейс JTAG-программирования (фактически состоит из нескольких физических и виртуальных регистров данных) используется для последовательного программирования через интерфейс JTAG. Цепь внутреннего сканирования и точки прерывания сканируемой цепи используются только встроенной системой отладки.

Порт доступа к функциям тестирования – TAP

JTAG-интерфейс задействует 4 вывода AVR-микроконтроллера. По JTAG-терминологии эти выводы в совокупности называются "Порт доступа к функциям тестирования" (TAP). В состав этого порта входят следующие сигналы:

TMS – Выбор режим тестирования. Данный вывод используется для навигации по цифровому автомату TAP-контроллера.
TCK: Синхронизация тестирования. JTAG-интерфейс работает синхронно по отношению TCK.
TDI: Тестовый ввод данных. Последовательный ввод данных сдвигом в регистр инструкции или регистр данных (цепи сканирования).
TDO: Тестовый вывод данных. Последовательный вывод данных из регистра инструкции или регистра данных.

По стандарту IEEE 1149.1 также определен опциональный TAP-сигнал: TRST – сброс тестирования, который у AVR-микроконтроллеров отсутствует.

Если конфигурационный бит JTAGEN незапрограммирован, то четыре TAP-вывода выполняют функции обычного порта ввода-вывода, а TAP-контроллер находится в состоянии сброса. Если же бит JTAGEN запрограммирован, а также сброшен бит JTD в регистре MCUCSR, то к входным сигналам порта TAP подключаются внутренние подтягивающие к плюсу питания резисторы и

разрешается работа интерфейса JTAG для граничного сканирования и программирования. В тех случаях, когда TAP-контроллер не выполняет сдвиг данных, выход порта TAP (вывод TDO) находится в третьем состоянии и, поэтому, должен быть подключен к внешнему подтягивающему резистору или к другому аппаратному компоненту, который содержит свой подтягивающий резистор (например, вход TDI следующего устройства в цепи сканирования). В состоянии поставки конфигурационный бит JTAGEN запрограммирован. Системой встроенной отладки в дополнении к сигналам интерфейса JTAG используется вывод RESET, состояние которого оценивает отладчик для определения возникновения условия внешнего сброса. Кроме того, отладчик может установить низкий уровень на выводе RESET, поэтому, необходимо следить, чтобы источник внешнего сброса в исполнительном каскаде содержал только открытый коллектор (сток).

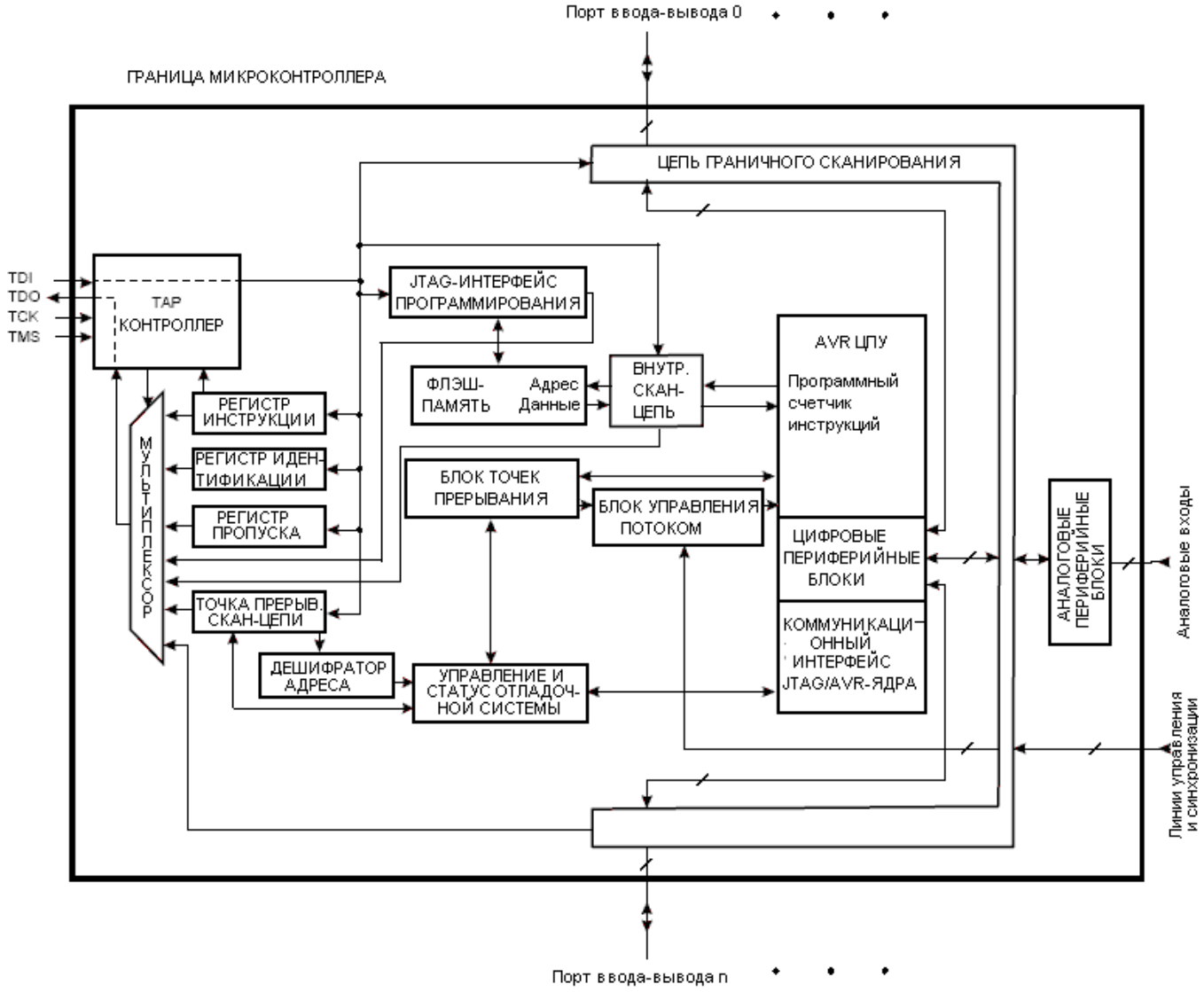


Рисунок 120 – Функциональная схема интерфейса JTAG и отладочной системы

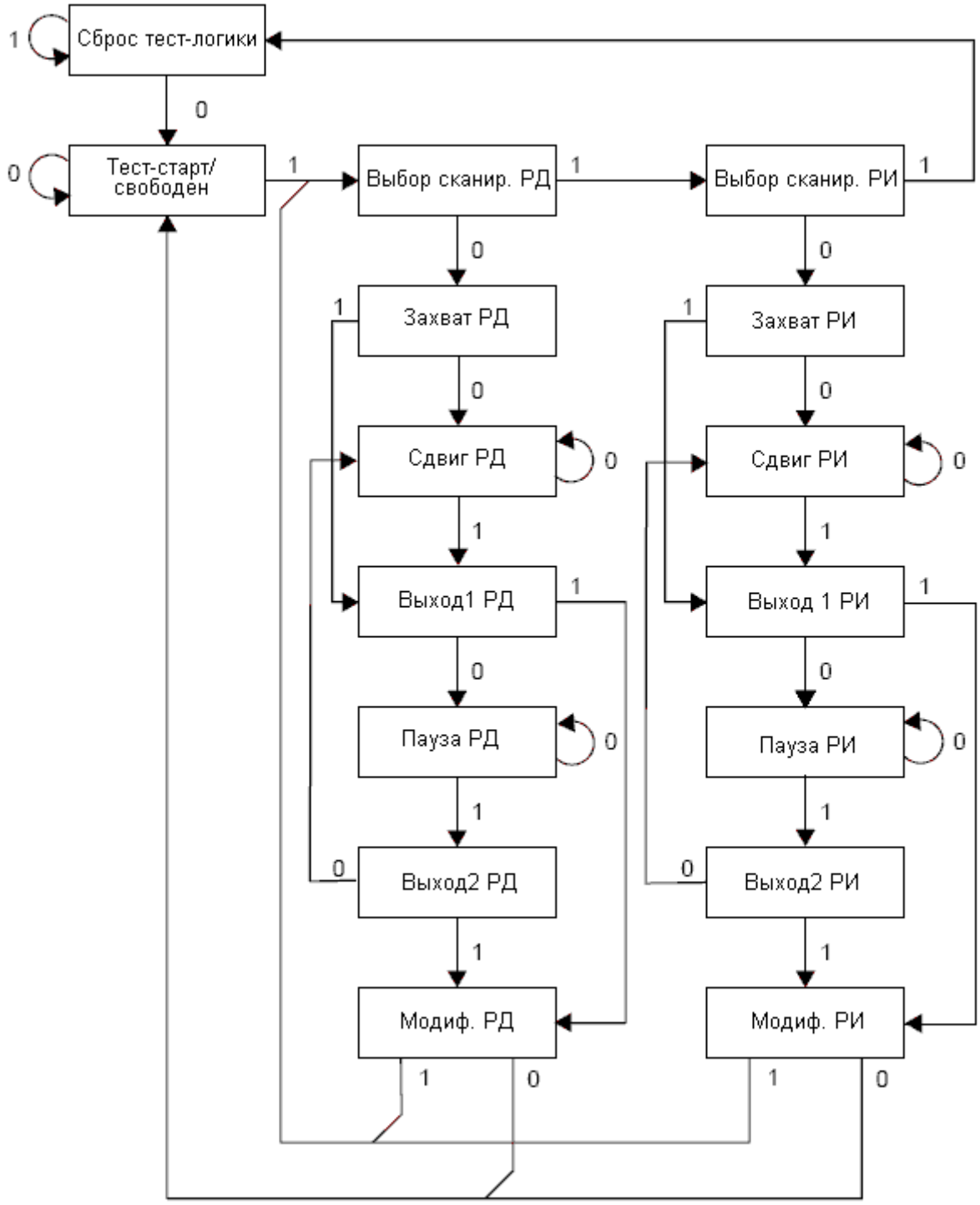


Рисунок 121 – Граф-автомат TAP-контроллера

TAP-контроллер

TAP-контроллер – цифровой автомат с 16 состояниями, который управляет работой схемой граничного сканирования, схемой JTAG-программирования или встроенной системой отладки. Граф-автомат, представленный на рисунке 121, зависит от сигнала TMS (его значение показывается рядом с переходом на другое состояние) в момент возникновения нарастающего фронта на TCK. Исходным состоянием после сброса при подаче питания является СБРОС ТЕСТ-ЛОГИКИ.

Все сдвиговые регистры выполняют первым ввод/вывод младших разрядов. Допустим, что текущим состоянием является ТЕСТ-СТАРТ/СВОБОДЕН, тогда типичным сценарием использования интерфейса JTAG является:

1. Формирование на входе TMS последовательности 1, 1, 0, 0 синхронно с нарастающим фронтом TCK для перевода в состояние сдвига регистра инструкции– СДВИГ ИР. Находясь в этом состоянии, выполняется сдвиг 4 разрядов JTAG-инструкции в регистр инструкции с входа TDI синхронно с нарастающим фронтом TCK. На входе TMS должен присутствовать низкий логический уровень в процессе ввода 3 младших разрядов для того, чтобы остаться в состоянии Сдвиг РИ. Старший разряд инструкции вводится сдвигом при выходе из данного состояния установкой лог. 1 на TMS. В процессе ввода инструкции через вывод TDI на выходе TDO выводится сдвигом код 0x01, означающий нахождение в состоянии СДВИГ РИ. JTAG-инструкция выбирает специфический регистр данных в качестве сдвигового регистра между выводами TDI и TDO, а также управляет схемой, связанной с регистром данных.
2. Формирование на входе TMS последовательности 1, 1, 0 для повторного ввода в состояние ТЕСТ-СТАРТ/СВОБОДЕН. В состоянии МОДИФ. ИР инструкция защелкивается на параллельном выходе сдвигового регистра. Состояния ВЫХОД1 ИР, ПАУЗА ИР и ВЫХОД2 ИР используются только для навигации по цифровому автомату.
3. Формирование на входе TMS последовательности 1, 0, 0 синхронного с нарастающими фронтами TCK для ввода состояния сдвига регистра данных – Сдвиг РД. Находясь в этом состоянии, загружается регистр данных (какой именно регистр определяется действующей инструкцией в регистре инструкции) с входа TDI нарастающим фронтом TCK. Для удержания состояния СДВИГ РД вход TMS должен оставаться в низком состоянии в процессе ввода всех разрядов за исключением последнего (старшего). Старший разряд вводится при выходе из этого состояния при установке высокого уровня на входе TMS. Во время последовательной записи в сдвиговый регистр с вывода TDI параллельные входы регистра данных захватываются в состоянии ЗАХВАТ РД и передаются сдвигом на выводе TDO.
4. Формирование на входе TMS последовательности 1, 1, 0 для повторного ввода состояния ТЕСТ-СТАРТ/СВОБОДЕН. Если у выбранного регистра данных параллельный выход содержит защелку, то защелкивание происходит в состоянии МОДИФ. РД. Состояния ВЫХОД1 РД, ПАУЗА РД и ВЫХОД2 РД используются только для навигации по граф-автомату.

Как показано на граф-автомате состояние ТЕСТ-СТАРТ/СВОБОДЕН не должно вводиться между выбором JTAG-инструкции и использованием регистров данных, а также некоторые JTAG-инструкции активизируют функции, которые работают даже в состоянии ТЕСТ-СТАРТ/СВОБОДЕН, что делает это состояние неприемлемым в качестве свободного состояния.

Прим.: Независимо от исходного состояния TAP-контроллера перейти в состояние СБРОС ТЕСТ-ЛОГИКИ можно путем удержания входа TMS в высоком состоянии в течение 5 тактов TCK.

Более детальная информация по техническим требованиям к JTAG приведена в специализированной литературе (см. "Список литературы").

Использование цепи граничного сканирования

Полное описание возможностей граничного сканирования дано в разделе “Граничное сканирование с соответствии с IEEE 1149.1 (JTAG)”.

Использование встроенной отладочной системы

Как показано на рисунке 120 аппаратная часть отладочной системы состоит из:

Скан-цепь на интерфейсе между внутренним AVR ЦПУ внутренними периферийными блоками
Блок точек прерывания
Коммуникационный интерфейс между ЦПУ и JTAG-системой

Все операции чтения или модификация/запись, необходимые для работы отладчика, выполнены путем применения AVR-инструкций через внутреннюю скан-цепь AVR ЦПУ. ЦПУ отправляет результат в память ввода-вывода по указанному адресу, которая является частью коммуникационного интерфейса между ЦПУ и JTAG-системой.

Блок точек прерывания поддерживает прерывания по изменению программного потока, пошаговое прерывание, две точки прерывания в памяти программ и две комбинированных точки прерывания. Вместе, четыре точки прерывания могут конфигурироваться как:

4 раздельных точки прерывания памяти программ
3 раздельных точки прерывания памяти программ + 1 точка прерывания памяти данных
2 раздельных точки прерывания памяти программ + 2 раздельных точки прерывания памяти данных
2 раздельных точки прерывания памяти программ + 1 точка прерывания памяти программ с маской ("множество точек прерывания")
2 раздельных точки прерывания памяти программ + 1 точка прерывания памяти данных с маской для задания множества точек прерываний

Однако, отладчик, например AVR Studio, может использовать только ограниченный набор этих возможностей, что снижает гибкость отладки для конечного пользователя. Перечень специфических JTAG-инструкций встроенной отладки представлен в разделе “Специальные JTAG-инструкции встроенной отладочной системы”.

Для разрешения работы порта доступа к функциям тестирования (TAP) необходимо запрограммировать конфигурационный бит JTAGEN. Кроме того, работа встроенной отладочной системы возможна только, если запрограммирован конфигурационный бит OCDEN и нет установленных бит защиты программы. В целях безопасности работа отладочной системы блокируется, если установлен какой-либо из бит защиты программы. В противном случае, отладочная система могла бы служить способом считывания защищенной разработчиком программы.

AVR Studio предоставляет разработчику полное управление выполнением программы AVR-микроконтроллера за счет использования встроенной в микроконтроллер отладочной системы, внутрисхемного эмулятора или автономного симулятора AVR-инструкций. AVR Studio поддерживает выполнение программ на Ассемблере, откомпилированных ассемблером корпорации Atmel, а также программ на Си, откомпилированных программами сторонних производителей.

AVR Studio работает под операционными системами Microsoft Windows 95/98/2000 и Windows NT. В данном документе представлены только общие черты программы AVR Studio, а полное описание представлено в "Руководство пользователя по AVR Studio" (AVR Studio User Guide).

Все необходимые команды могут быть выполнены в AVR Studio, как на уровне исходного кода, так и на дезассемблированном уровне. Пользователь может запустить программу на исполнение, выполнить программу пошагово как с переходом к обработке подпрограмм, так и без, выполнить шаг назад, выполнить программу до позиции курсора, остановить программу, а также сбросить текущий сеанс выполнения программы (эквивалентно сбросу микроконтроллера). Кроме того, пользователь может установить неограниченное число точек прерывания по адресу памяти программ (используя инструкцию BREAK) и до двух точек прерываний по обращению к адресам памяти данных, которые альтернативно могут образовывать маску адресов памяти данных для прерывания выполнения программы.

Специальные JTAG-инструкции встроенной отладочной системы

Встроенная отладочная система поддерживает специальные инструкции, которые известны только внутри корпорации ATMEL и сторонним производителям отладочных средств. Для справки ниже приведены коды операций.

PRIVATE0; $8 специальная JTAG-инструкция для обращения к встроенной отладочной системе.

PRIVATE1; $9 специальная JTAG-инструкция для обращения к встроенной отладочной системе.

PRIVATE2; $A специальная JTAG-инструкция для обращения к встроенной отладочной системе.

PRIVATE3; $B специальная JTAG-инструкция для обращения к встроенной отладочной системе.

Регистр встроенной отладочной системы – OCDR

Разряд	7	6	5	4	3	2	1	0	
	Ст.разр. IDRD	-	-	-	-	-	-	Мл.разр.	**OCDR**
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	

Модулятор выходов таймеров (OCM1C2)

Введение

Модулятор выходов таймеров (OCM) позволяет генерировать прямоугольные импульсы, промодулированные несущей частотой. Модулятор использует выходы канала сравнения С 16-разр. таймера-счетчика 1 и выход блока сравнения 8-разр. таймера-счетчика 2. Обратите внимание, что данная функция не поддерживается в режиме совместимости с ATmega103.

Если работа модулятора разрешена, то сигналы с выходов каналов сравнения объединяются в один в соответствии с рис. 72.

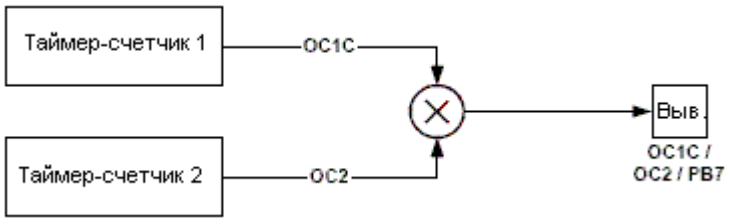


Рисунок 72. Принцип выполнения модуляции

Описание

Блоки сравнения 1С и 2 использует один и тот же вывод порта PB7 в качестве своего выхода. Выходы блоков сравнения (OC1C и OC2) блокируют обычную функцию регистра PORTB7 после разрешения работы одного из них (в т.ч., если COMnx1:0 не равны 0). После разрешения работы OC1C и OC2 автоматически разрешается работа модулятора.

Эквивалентная функциональная схема модулятора представлена на рисунке 73. Схема содержит часть блоков таймеров-счетчиков и схему выходного драйвера линии 7 порта B.

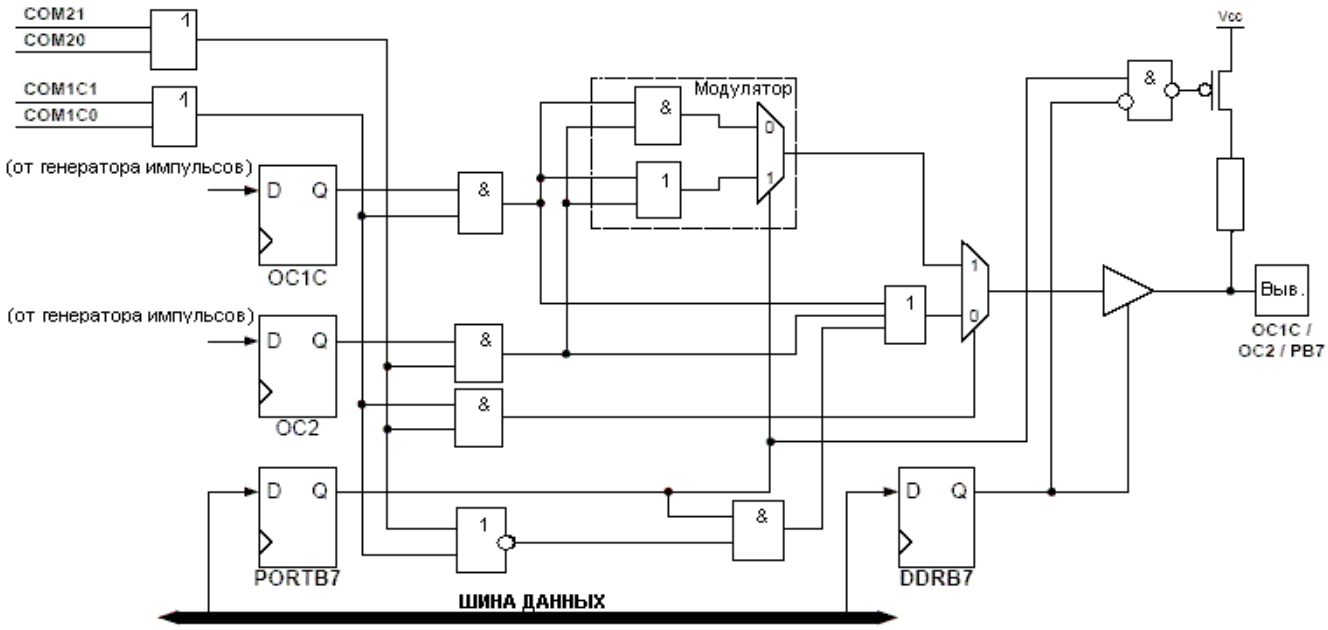


Рисунок 73. Эквивалентная функциональная схема модулятора выходов таймеров

После разрешения работы модулятора необходимо выбрать тип модуляции (лог. И или ИЛИ) с помощью регистра PORTB7. Обратите внимание, что DDRB7 управляет направлением независимо от установок бит COMnx1:0.

Пример временной диаграммы

Рисунок 74 иллюстрирует работу модулятора. В данном примере таймер-счетчик 1 настроен на работу в режиме быстрой ШИМ (без инверсии), а таймер-счетчик 2 генерирует импульсы в режиме CTC (сброс таймера при совпадении) с переключением выходного состояния выхода компаратора при совпадении (COMnx1:0 = 1).

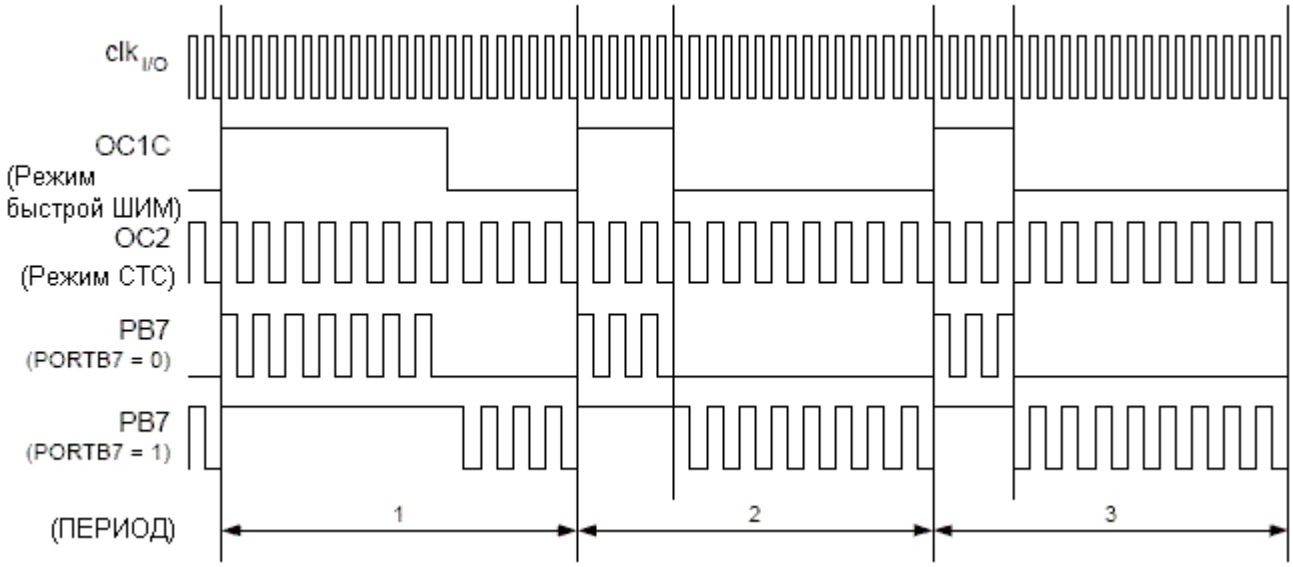


Рисунок 74. Временная диаграмма работы модулятора

В данном примере таймер-счетчик 2 генерирует несущий сигнал, а модулирующий сигнал генерируется каналом C блока сравнения таймера-счетчика 1.

Разрешающая способность ШИМ-сигнала (OC1C) снижается за счет модуляции. Коэффициент снижения эквивалентен числу тактовых импульсов системной синхронизации в течение одного периода несущего сигнала (OC2). В данном примере разрешение снижено с коэффициентом 2.

Причина снижения разрешения показана ни рисунке 74 во 2 и 3 периоде сигнала на выходе PB7, когда состояние PORTB7 равно нулю. Длительность единичного импульса 2-го периода (OC1C) на один такт дольше длительности единичного импульса 3-го периода, но форма сигнала на выводе PB7 одинакова на каждом из этих периодов.

Последовательный периферийный интерфейс - SPI

Интерфейс SPI позволяет организовать последовательную синхронную высокоскоростную передачу данных между ATmega128 и другим периферийным устройством или между несколькими AVR-микроконтроллерами.

Отличительные особенности интерфейса SPI в ATmega128:

Полнодуплексная, трехпроводная синхронная передача данных
Ведущая или подчиненная работа
Передача первым младшего или старшего 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)

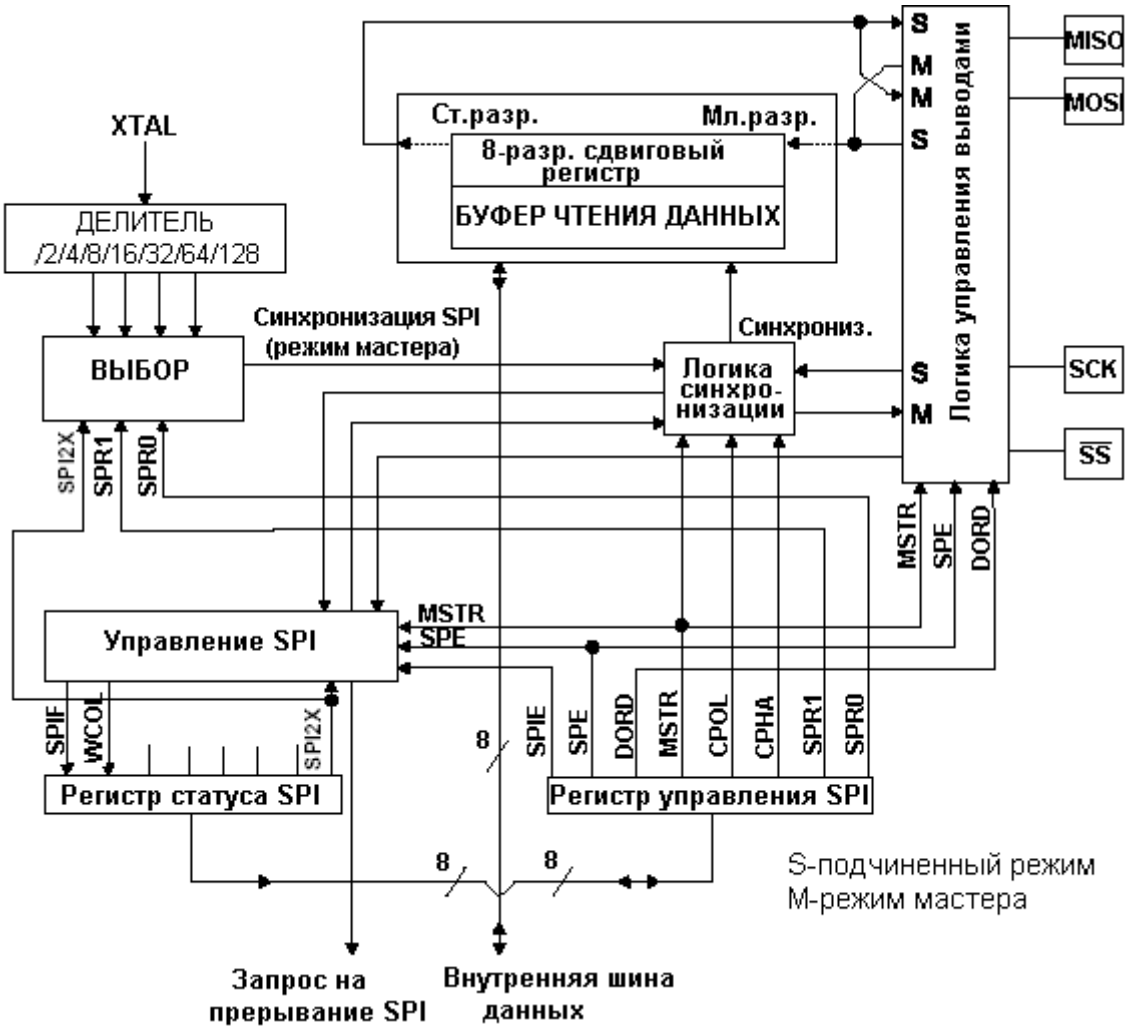


Рисунок 75. Функциональная схема интерфейса SPI

Прим.: Расположение выводов интерфейса SPI представлено на рис. 1 и в таблице 30.

Внешние соединения между ведущим (мастером) и подчиненным ЦПУ через интерфейс SPI показаны на рисунке 76. 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.

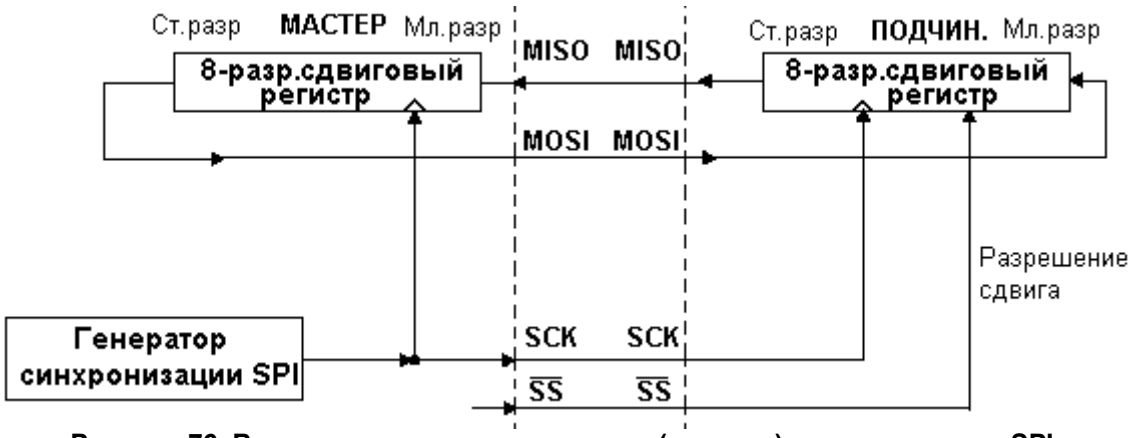


Рисунок 76. Внешнее соединение ведущего (мастера) и подчиненного SPI

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 необходимо использовать частоту синхронизации SPI не более fosc/4.

Если работа SPI разрешена, то разрешается альтернативное направление выводов MOSI, MISO, SCK и SS (см. табл. 69).

Таблица 69. 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.

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 (для ATmega128 DDRB). 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

Разряд	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 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 SPI.

Разряд 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.

Разряд 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.

Разряд 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.

Разряд 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. 77 и 78. Ниже обобщено функционирование CPOL:

Таблица 70. Результат действия CPOL

CPOL	Передний фронт	Задний фронт
0	Нарастающий	Спадающий
1	Спадающий	Нарастающий

Разряд 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 77 и 78. Действие CPHA подытожено ниже:

Таблица 71. Результат действия бита CPHA

CPHA	Передний фронт	Задний фронт
0	Выборка	Установка
1	Установка	Выборка

Разряды 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 72. 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

Разряд	7	6	5	4	3	2	1	0	
	SPIF	WCOL	-	-	-	-	-	SPI2X	SPSR
Чтение/запись	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 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).

Разряд 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.

Разряды 5..1 - Res: зарезервированные биты

В ATmega128 данные биты не используются и всегда считываются как 0.

Разряд 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 ATmega128 также используется для чтения или программирования памяти программ и ЭСППЗУ. См. также "Последовательное программирование".

Регистр данных SPI - SPDR

Разряд	7	6	5	4	3	2	1	0	
	Ст.разр.							Мл.разр.	SPDR
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	x	x	x	x	x	x	x	x	Неопред.

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 73, а их временные диаграммы показаны на рис. 77 и 78. 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. 70 и 71 и представить ее в следующем виде:

Таблица 73. 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

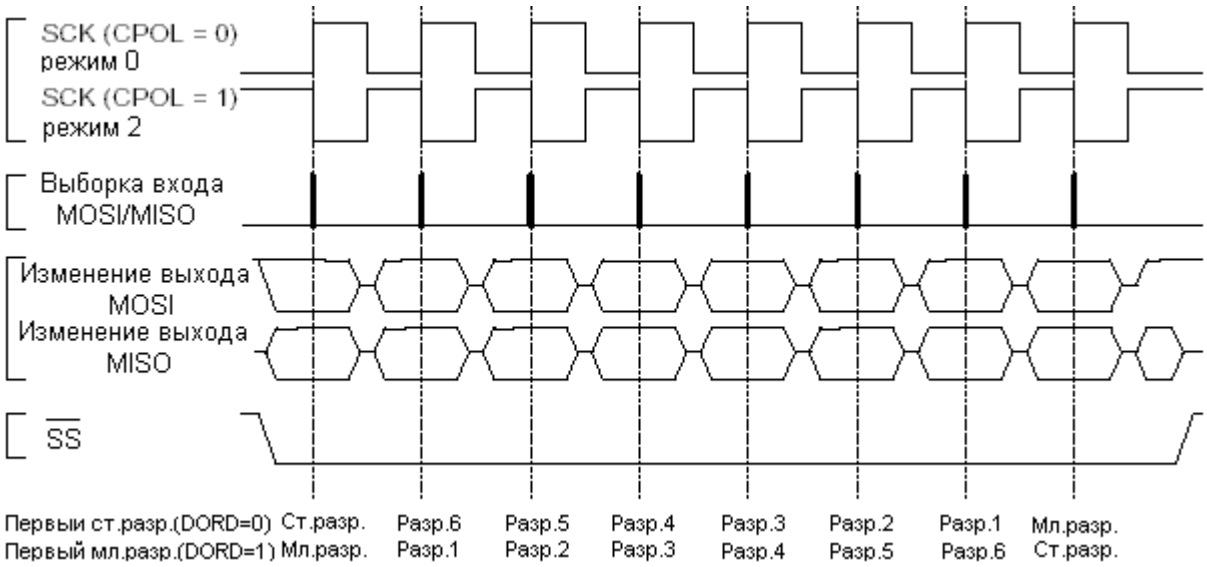


Рисунок 77. Формат передачи данных SPI с CPHA = 0

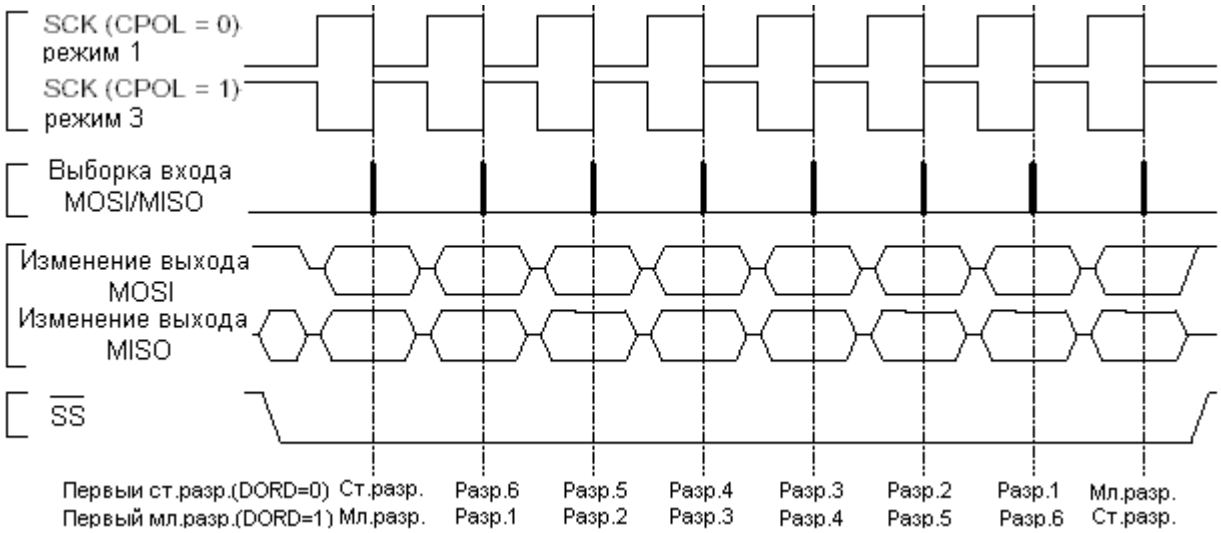


Рисунок 78. Формат передачи данных SPI с CPHA = 1

8-разр. таймер-счетчик 0 с функциями широтно-импульсной модуляции и асинхронного тактирования

Таймер-счетчик 0 - модуль многофункционального одноканального 8-разрядного таймера-счетчика с аппаратным выходом для генерации ШИМ-сигнала и встроенным асинхронным опциональным тактовым генератором, который оптимизирован под использование часового кварца (32768Гц) для асинхронного по отношению к системной синхронизации тактирования.

Основные отличительные особенности:

Одноканальный счетчик
Опциональный режим сброса таймера при совпадении (автоматическая перезагрузка)
Широтно-импульсная модуляция без генерации ложных импульсов при записи нового порога сравнения в OCR0 (двойная буферизация) и с фазовой коррекцией
Генератор частоты
10-разрядный предделитель тактовой частоты
Генерация прерываний по переполнению и выполнения условия сравнения (TOV0 и OCF0)
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. 34. 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 0".

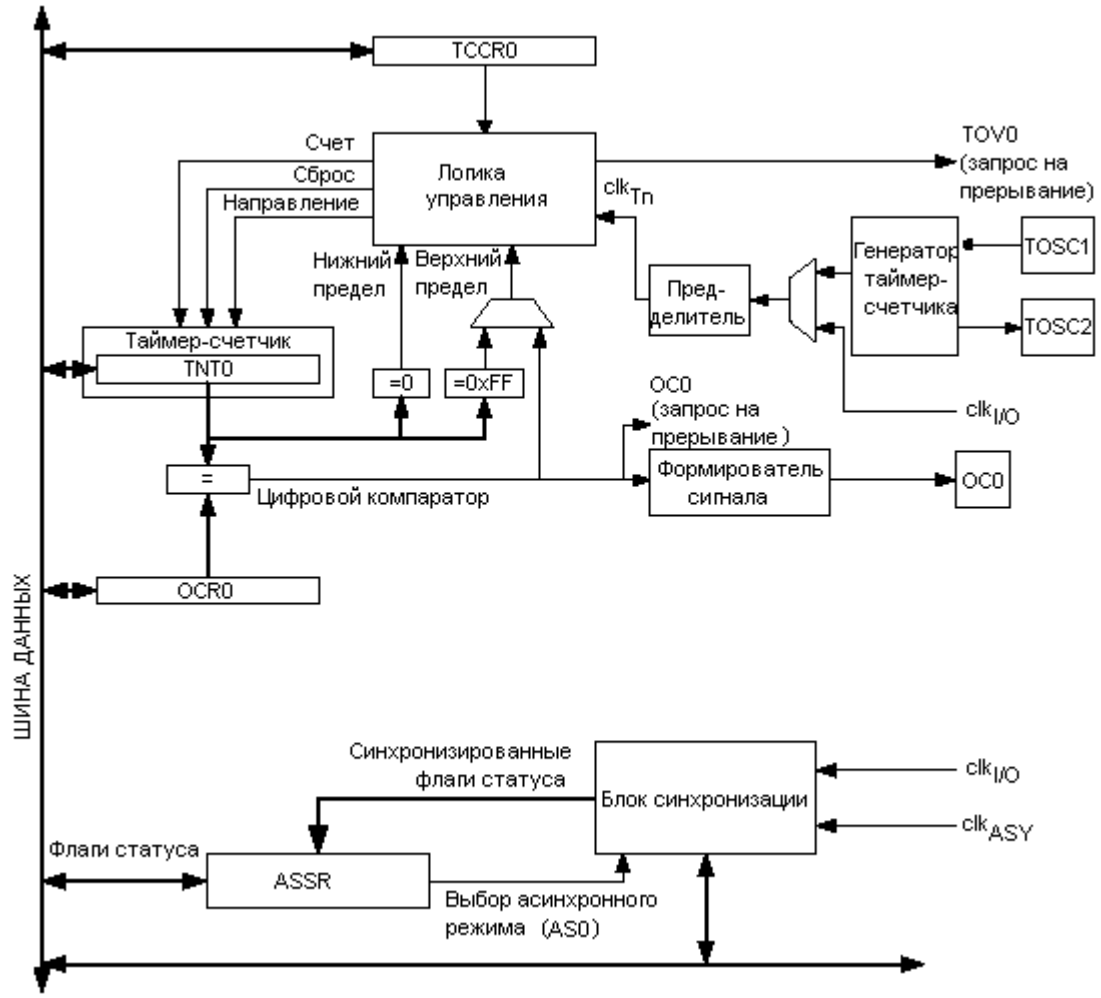


Рисунок 34. Функциональная схема 8-разр. таймера-счетчика

Регистры

Регистр таймера-счетчика (TCNT0) и регистр порога сравнения (OCR0) - 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 (clkT0).

Значение регистра порога сравнения с двойной буферизацией (OCR0) непрерывно сравнивается со значением таймера-счетчика. Результат сравнения может использоваться для генерации сигналов с ШИМ или прямоугольных импульсов переменной частоты на выводе OC0.

См. "Блок сравнения". Совпадение порога сравнения со значением таймера-счетчика приводит к установке флага результата сравнения (OCF0), который может использоваться для генерации запроса на прерывание по результату сравнения.

Определения

Некоторые определения и их сокращенные наименования, которые интенсивно используются в этом разделе, представлены в таблице 51.

Таблица 51. 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 OCR0.

Тактовые источники таймера-счетчика 0

Таймер-счетчик 0 может тактироваться внутренне синхронно или внешне асинхронно (по отношению к внутренней системной синхронизации). По умолчанию используется тактовый сигнал clkT0, эквивалентный тактовому сигналу микроконтроллера clkI/O. Если в бит AS0 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". Подробности по источникам синхронизации см. в разделе "Предделитель таймера-счетчика 0".

Блок счетчика

Основу 8-разр. таймера-счетчика 0 составляет программируемый двунаправленный счетчик. Рисунок 35 показывает функциональную схему счетчика и окружающих его элементов.

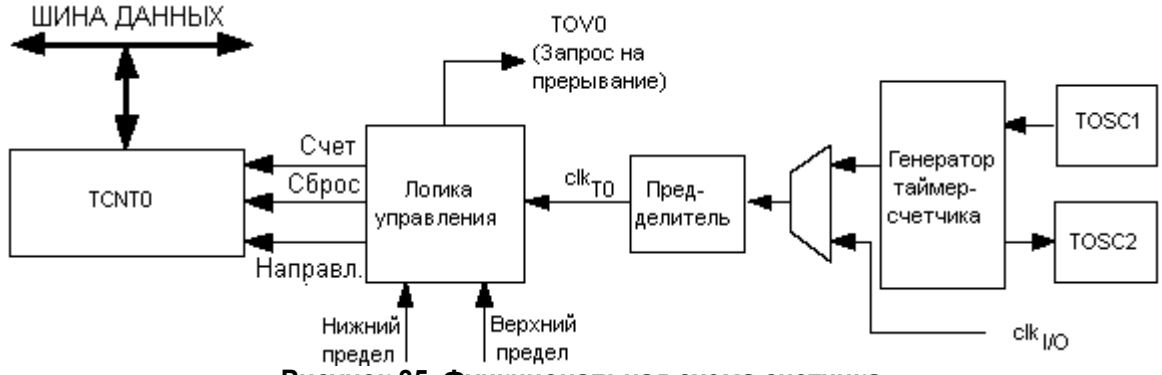


Рисунок 35. Функциональная схема счетчика

Описание сигналов (внутренние сигналы):

Счет - Инкрементирует или декрементирует TCNT0 на 1.
Направление - Задает направление счета: инкрементирование (+1, прямой счет) или декрементирование (-1, обратный счет).
Сброс - Сбрасывает содержимое TCNT0 (запись лог. 0 во все разряды).
clkT0 - Синхронизация таймера-счетчика.
Верхний предел - Задает максимальное значение, которое может достигнуть TCNT0.
Нижний предел - Задает минимальное значение, которое может достигнуть TCNT0 (ноль).

В зависимости от выбранного режима работы счетчик сбрасывается, инкрементируется или декрементируется на каждом такте синхронизации (clkT0). Тактовый сигнал clkT0 может быть

внутренним или внешним, а его частота выбирается с помощью бит выбора частоты синхронизации CS02-CS00. Если источник синхронизации не задан (CS02-CS00 = 0b000), 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.

Последовательность счета определяется установкой бит WGM01 и WGM00, расположенных в регистре управления таймером-счетчиком (TCCR0). Имеется точная связь между поведением счетчика (алгоритмом счета) и генерируемой на выходе OC0 формы сигнала. Более подробно об алгоритмах счета и генерации импульсов написано в "Режимы работы".

Флаг переполнения таймера-счетчика (TOV0) устанавливается в соответствии с режимом работы, который выбирается битами WGM01, WGM00. Бит TOV0 может использоваться для генерации прерывания ЦПУ.

Блок сравнения

8-разрядный цифровой компаратор непрерывно выполняет сравнение содержимого регистра таймера-счетчика TCNT0 с регистром порога сравнения OCR0. Всякий раз, когда значение TCNT0 совпадает со значением OCR0 компаратор устанавливает флаг совпадения OCF0 следующим тактом синхронизации таймера. Если разрешено прерывание битом OCIE0 = 1, то установка флага совпадения вызывает запрос на прерывание. Флаг OCF0 автоматически сбрасывается во время выполнения процедуры обработки прерывания. Альтернативно, флаг OCF0 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 WGM01, WGM00 и битами задания режима формирования выходного сигнала (COM01, COM00). Верхний и нижний пределы счета используются в некоторых режимах работы для выполнения специальных действий (см. "Режимы работы таймера -счетчика 0"). На рисунке 36 приведена функциональная схема блока сравнения.

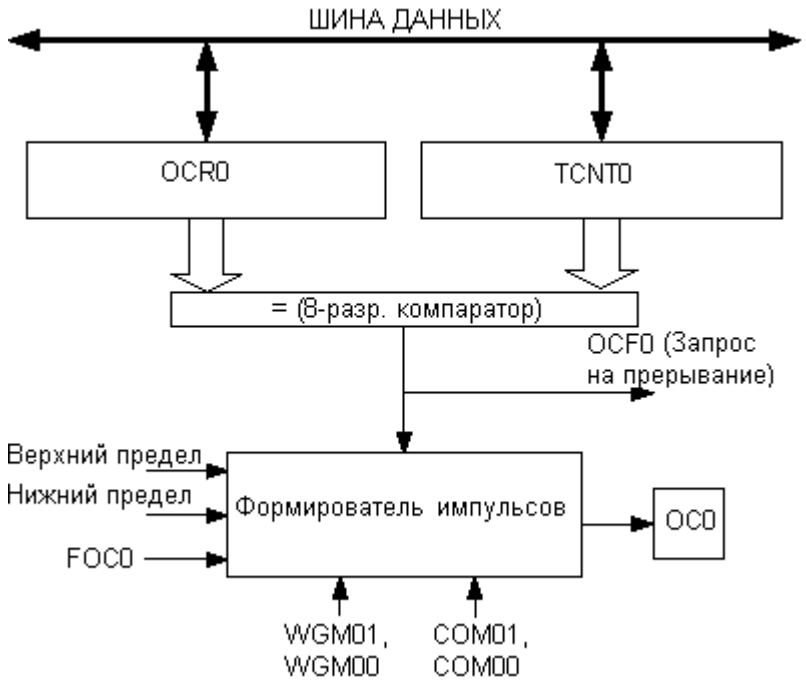


Рисунок 36. Функциональная схема блока сравнения

Регистр OCR0 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 OCR0 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 OCR0 может показаться сложным, но это выполнено не случайно. После разрешения двойной буферизации ЦПУ осуществляет доступ к буферному регистру OCR0, а после отключения - непосредственно адресуется к регистру OCR0.

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 FOC0. Принудительная установка результата сравнения компаратора не приводит к установке флага OCF0 или сбросу/перезагрузке таймера, но влияет на состояние вывода OC0, который будет устанавливаться, сбрасываться или переключаться (инвертироваться) в зависимости от выбранной установки бит COM01, COM00.

Результат сравнения блокируется записью в TCNT0

Если ЦПУ осуществляет запись в регистр TCNT0, то результат сравнения будет игнорироваться на следующем такте синхронизации таймера, даже если таймер остановлен. Данная функция позволяет установить в регистре OCR0 то же значение, что и в TCNT0 без генерации запроса на прерывание, если разрешено тактирование таймера-счетчика.

Использование блока сравнения

Поскольку запись в TCNT0 блокирует любые действия по результату сравнения на один такт синхронизации таймера независимо от режима работы, то при изменении TCNT0 при использовании канала сравнения (независимо работает синхронизация таймера или нет) необходимо учесть следующие особенности. Если в регистр TCNT0 записано значение равное OCR0, то игнорирование совпадения приведет к генерации некорректной формы сигнала. По аналогии следует избегать записи в TCNT0 значения равного нижнему пределу (0x00), если счетчик работает как вычитающий.

Установка OC0 должна быть выполнена перед настройкой линии ввода-вывода на вывод в регистре направления данных. Самый легкий путь установки значения OC0 - использование бита принудительной установки результата сравнения (FOC0) в нормальном режиме. Регистр OC0 сохраняет его значение, даже если происходит изменение режима работы таймера.

Учтите, что биты COM01, COM00 не содержат схемы двойной буферизации и на любые изменения реагируют мгновенно.

Блок формирования выходного сигнала

Биты задания режима формирования выходного сигнала (COM01:0) имеют двойное назначение. С одной стороны биты COM01, COM00 используются формирователем сигнала и определяют какое логическое состояние должно быть на выходе OC0 при возникновении следующего совпадения. С другой стороны, биты COM01, COM00 используются для разрешения/запрета альтернативной функции вывода порта OC0. На рисунке 37 представлена упрощенная логическая схема, на которую воздействуют биты COM01, COM00. На рисунке показаны только те регистры управления портом ввода-вывода (DDR и PORT), на которые оказывает действие биты COM01, COM00.

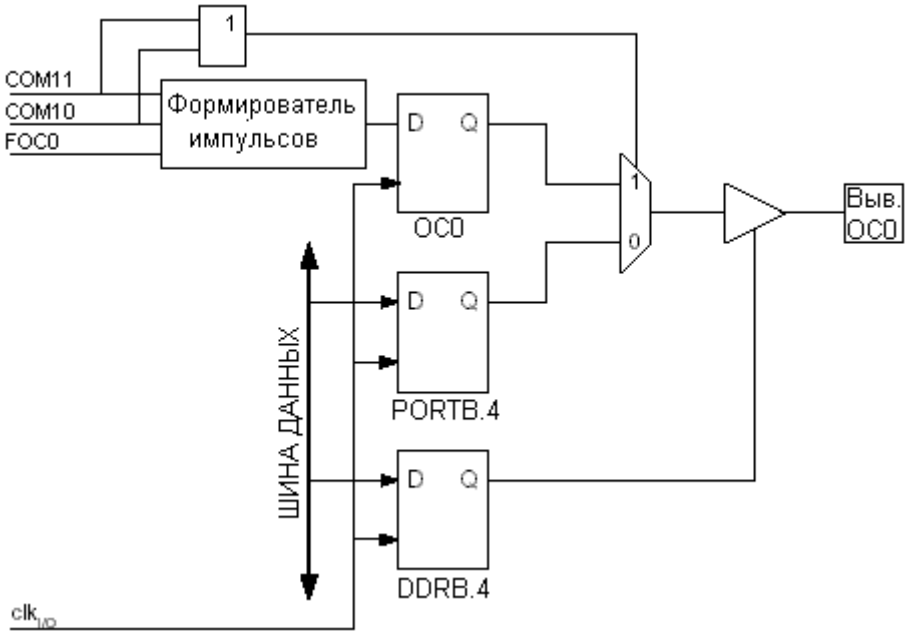


Рисунок 37. Схема блока формирования выходного сигнала

Функция линии универсального порта ввода-вывода заменяется на функцию выхода формирователя сигнала OC0, если хотя бы один из бит COM01, COM00 установлен (логика ИЛИ). Однако, управлением направлением вывода OC0 (вход или выход) в этом случае остается за соответствующим битом регистра направления данных порта B (DDRB.4). Чтобы значение регистра OC0 присутствовало на выводе OC0 необходимо настроить данную линию на вывод (установить бит DDRB.4). Управление вводом альтернативной функции не зависит от режима генерации сигнала.

Схемотехника выходной логики позволяет инициализировать состояние регистра OC0 перед разрешением настройки вывода OC0 в качестве выхода. Обратите внимание, что в некоторых режимах работах имеются зарезервированные состояния бит COM01, COM00. См. "Описание регистров 8-разр. таймера-счетчика 0".

Режимы генерации сигнала

Значение бит COM01:0 задает различные режимы формирования выходного сигнала, которые в свою очередь зависят от выбранного режима работы таймера. Таймер-счетчик 0 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 OC0 при выполнении условия сравнения, если оба бита COM01, COM00 равны нулю. В таблице 53 описано действие различных установок этих бит для режимов без ШИМ. Аналогичная информация для режима с быстрой ШИМ приведена в таблице 54, а для ШИМ с фазовой коррекцией в таблице 55.

После установки бит COM01, COM00 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 FOC0.

Режимы работы таймера-счетчика 0

Режим работы таймера, в т.ч. поведение таймера-счетчика и связанного с ним выхода формирователя сигнала, задается комбинацией бит, задающих режим работы таймера (WGM01, WGM00) и режим формирования выходного сигнала (COM01, COM00). 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 COM01, COM00 позволяют включить/отключить инверсию на генерируемом ШИМ-выходе (т.е. выбрать ШИМ с инверсией или ШИМ без инверсии). Для режимов без ШИМ биты COM01:0 определяют какое действие необходимо выполнить при выполнении условия сравнения: сбросить, установить или инвертировать выход (см. также "Блок формирования выходного сигнала").

Более подробная информация по временным диаграммам работы таймера-счетчика 0 приведена в разделе "Временные диаграммы таймера-счетчика 0".

Нормальный режим работы

Самым простым режимом работы является нормальный режим (WGM01, WGM00 = 0b00). 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 TOV0 будет установлен на том же такте синхронизации, когда TCNT0 примет нулевое значение. Фактически, флаг переполнения TOV0 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 TOV0 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 OC0 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 (WGM01, WGM00 =0b10) регистр OCR0 используется для задания разрешающей способности счетчика. Если задан режим CTC и значение счетчика (TCNT0) совпадает со значением регистра OCR0, то счетчик обнуляется (TCNT0=0). Таким образом, OCR0 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 38. Значение счетчика (TCNT0) инкрементируется до тех пор, пока оно не станет равным значению в OCR0, после чего счетчик (TCNT0) обнулится.

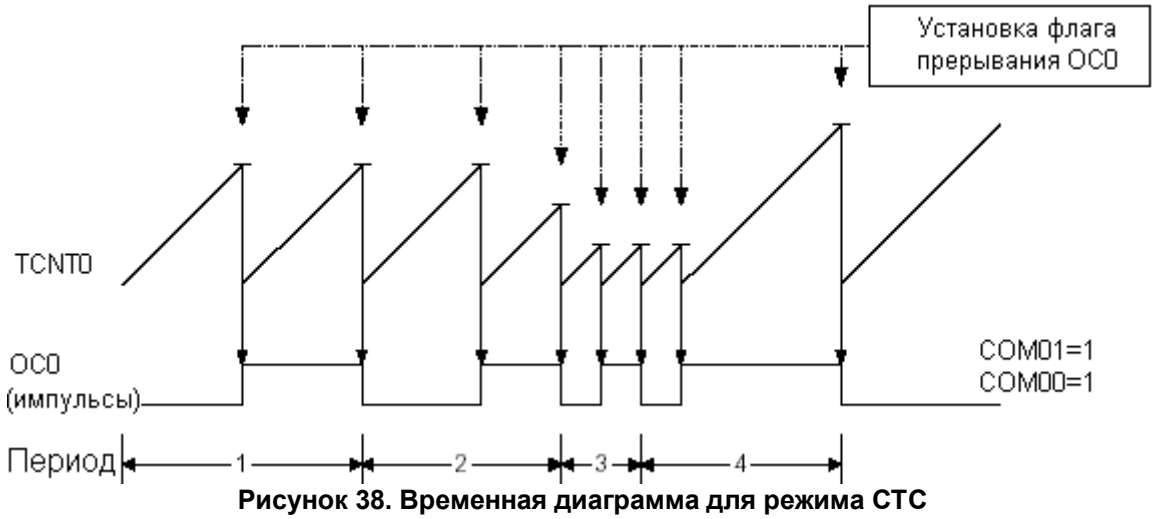


Рисунок 38. Временная диаграмма для режима CTC

С помощью флага OCF0 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 OCR0, меньше текущего значения TCNT0, то сброс счетчика по условию совпадения наступит, когда он

достигнет максимального значения (0xFF), затем перейдет в исходное состояние 0x00 и достигнет нового значения OCR0.

Для генерации сигнала в режиме CTC выход компаратора OC0 может использоваться для изменения логического уровня при каждом совпадении, для чего необходимо задать режим переключения (COM01, COM00 = 0b01). Значение OC0 будет присутствовать на выводе порта, только если для данного вывода задано выходное направление. Максимальная частота генерируемого сигнала равна fOC0 = fclk_I/O/2, если OCR0 = 0x00. Для других значений OCR0 частоту генерируемого сигнала можно определить по формуле:

$$f_{OC0} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCR0)},$$

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

Также как и для нормального режима работы, флаг TOV0 устанавливается на том же такте таймера, когда его значение изменяется с 0xFF на 0x00.

Режим быстрой ШИМ

Режим быстрой широтно-импульсной модуляции (WGM01, WGM00 = 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 TCNT0 и OCR0 сигнал OC0 сбрасывается, а при достижении нижнего предела счета устанавливается. Если задан инвертирующий режим, то выход OC0 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. 39. Значение TCNT0 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 TCNT0, где совпадают значения OCR0 и TCNT0.

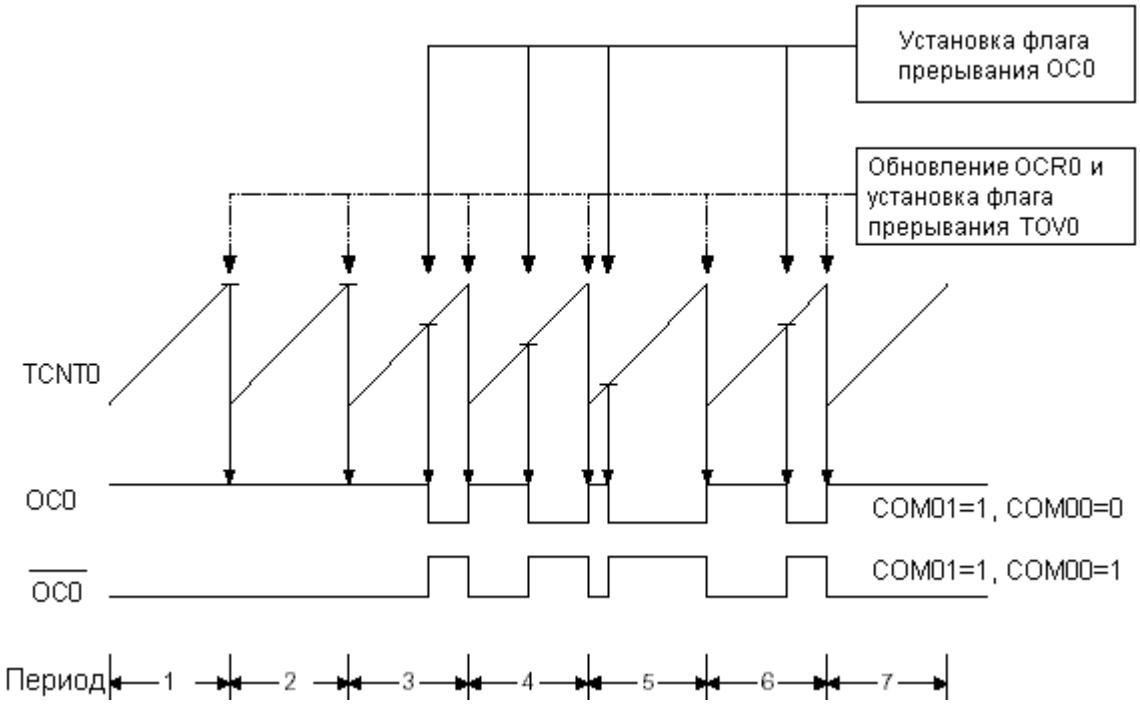


Рисунок 39. Временная диаграмма режима быстрой ШИМ

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 OC0. Если COM01:0=0b10, то выходной ШИМ-сигнал не инвертируется, а если COM01:0=0b11, то инвертируется (см. таблицу 54). Фактическое значение OC0 будет присутствовать на выводе порта, если для данного вывода задано выходное направление. ШИМ-сигнал генерируется путем установки (или сброса) сигнала OC0 при совпадении значений OCR0 и TCNT0 и сброса (или установки) сигнала OC0, если счетчик сбрасывается (состояние счетчика изменяется с максимального значения на нижний предел счета).

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

$$f_{OC0} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCR0)},$$

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

Задание предельных значений регистра OCR0 связано с особыми случаями в генерации ШИМ-сигнала в режиме быстрой ШИМ. Если в регистр OCR0 записать значение равное нижнему пределу счета, то через каждые 256 тактов синхронизации таймера на выходе будет генерироваться импульс короткой длительности. Если установить значение OCR0 равным максимальному значению, то выход будет иметь постоянный логический уровень 1 или 0 (зависит от заданного режима выхода битами COM01:0).

Генерация меандра (50%-ое заполнение импульсов) в данном режиме возможна, если задать режим переключения (инвертирование состояния) выхода OC0 при каждом совпадении (COM01, COM00 = 0b01). Если в регистр OCR0 записать ноль, то будет генерироваться максимальная частота foc0 = fclk_I/O/2. Данная функция похожа на переключение OC0 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 (WGM01, WGM00 = 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 OC0 сбрасывается/устанавливается при совпадении значений TCNT0 и OCR0 во время прямого/обратного счета. Если задан инвертирующий режим выхода, то, наоборот, во время прямого счета происходит установка, а во время обратного - сброс выхода OC0. 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 TCNT0 остается равным максимальному значению в течение одного такта синхронизации таймера. На рисунке 40 показана временная диаграмма работы таймера-счетчика для режима ШИМ ФК. Значение TCNT0 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 TCNT0, где совпадают значения OCR0 и TCNT0.

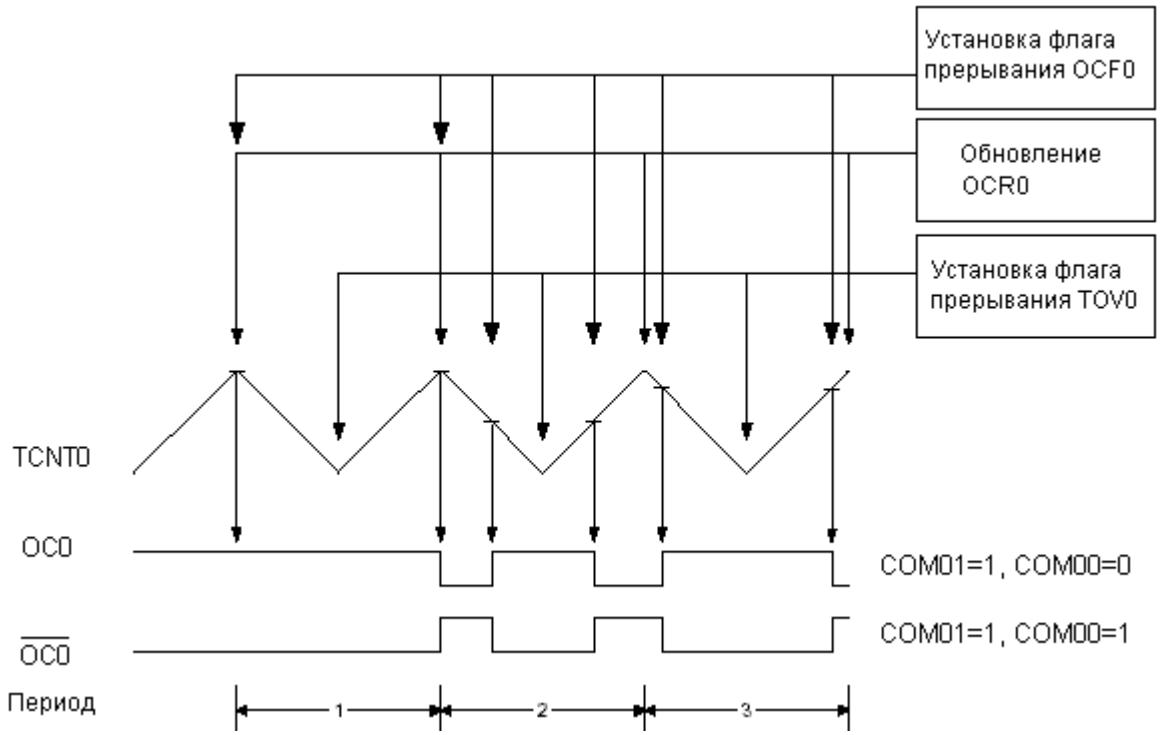


Рисунок 40. Временная диаграмма режима ШИМ ФК

Флаг переполнения таймера-счетчика (TOV0) устанавливается при достижении счетчиком нижнего предела и может использоваться для генерации прерывания.

В режиме ШИМ ФК блок сравнения позволяет генерировать ШИМ-сигналы на выводе OC0. Установка бит COM01,COM00 = 0b10 соответствует неинвертированному режиму генерации ШИМ-сигнала, а COM01, COM00 = 0b11 - инвертированному (см. табл. 55). Фактическое значение OC0 будет присутствовать на соответствующем выводе порта, если для него задано выходное направление. ШИМ-сигнал генерируется путем сброса (установки) сигнала OC0 при совпадении OCR0 и TCNT0 во время прямого счета и путем установки (сброса) сигнала OC0 при совпадении OCR0 и TCNT0 во время обратного счета. Результирующая частота ШИМ-сигнала в режиме ШИМ ФК может быть вычислена по следующему выражению:

$$f_{OC0\,ШИМ} = \frac{f_{clk\_I/O}}{N \cdot 256},$$

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

Задание предельных значений регистра OCR0 связано с особыми случаями генерации ШИМ-сигнала в режиме ШИМ ФК. Если значение OCR0 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 40 сигнал OC0 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:

OCR0 изменяет свое значение с максимального (0xFF) на другое (неравное 0xFF) (см. рис. 40). Если значение OCR0 равно максимальному, то вывод OC0 останется на прежнем уровне как результат совпадения во время обратного счета. Для гарантирования симметричности относительного нижнего предела (0x00) значение OC0 при максимальном значении счетчика должно соответствовать результату совпадении при прямом счете. Таймер начинает счет с большего значения по сравнению с OCR0 и по этой причине не возникает совпадение и, следовательно, изменение OC0, которое должно было возникнуть во время прямого счета.

Временные диаграммы таймера-счетчика 0

На рисунке 41 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 42-44 показаны аналогичные временные диаграммы, но с разрешенным предделением тактового сигнала. На рисунках также иллюстрируются моменты установки флагов прерываний, в т.ч. флаг OCF0 для всех режимов, кроме CTC, (рисунок 43) и для режима CTC (рисунок 44). На рисунке 44 также показан момент сброса таймера-счетчика TCNT0.

Представленные диаграммы соответствуют синхронному по отношению к системной частоте режиму тактирования таймера-счетчика. Однако, они будут полностью соответствовать асинхронному режиму работы, если заменить clkI/O на сигнал генератора таймера-счетчика.

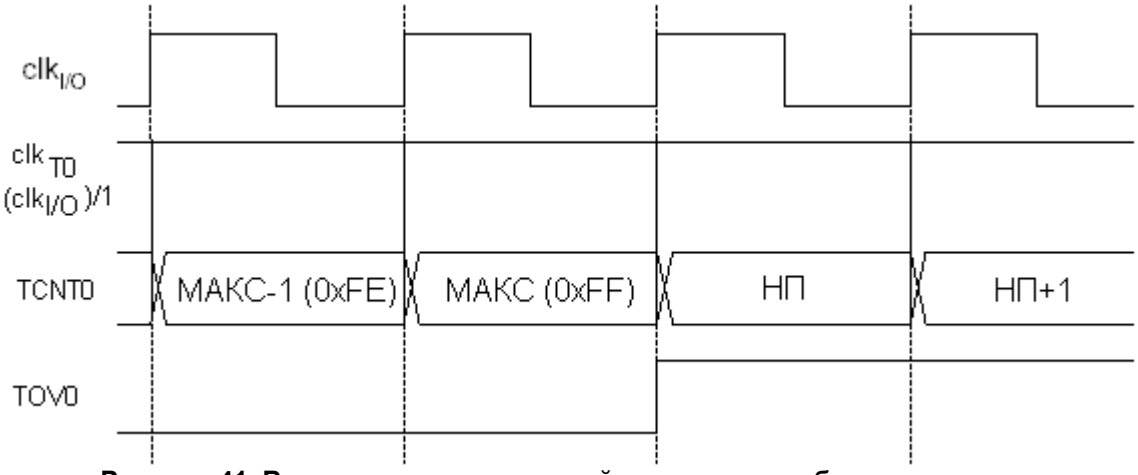


Рисунок 41. Временная диаграмма таймера-счетчика без предделения

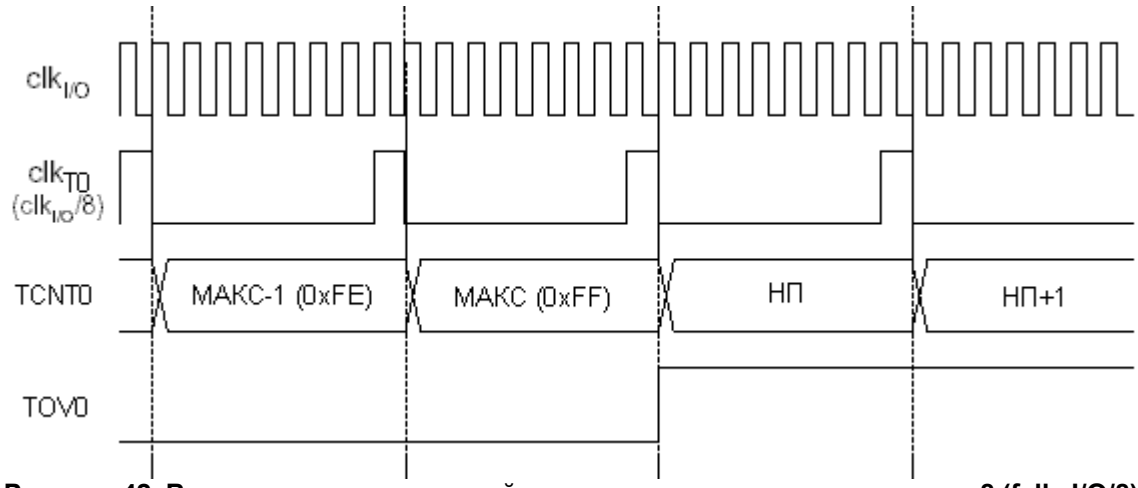


Рисунок 42. Временная диаграмма таймера-счетчика с предделением на 8 (fclk_I/O/8)

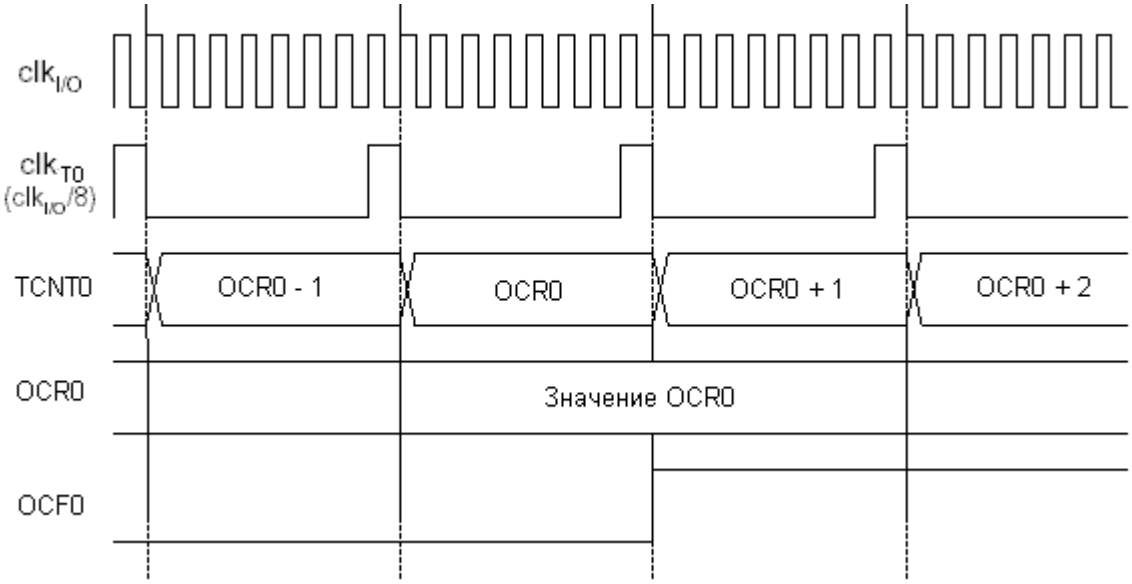


Рисунок 43. Временная диаграмма таймера-счетчика с установкой флага OCF0 и предделением на 8 (fclk_I/O/8)

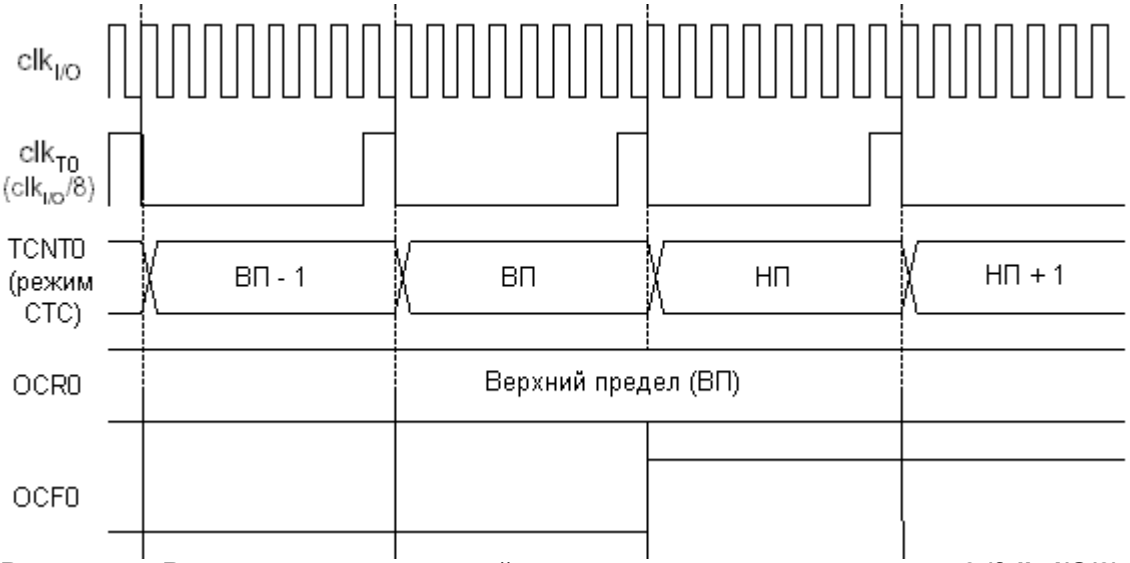


Рисунок 44. Временная диаграмма таймер-счетчика с предделением на 8 (fclk_I/O/8) в режиме сброса при совпадении

Описание регистров 8-разрядного таймера-счетчика 0

Регистр управления таймером-счетчиком 0 - TCCR0

Разряд	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Чтение/запись	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 7 - FOC0: Принудительная установка результата сравнения

Функция бита FOC0 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 TCCR0 в позиции данного бита указывать лог. 0, если таймер работает в одном из режимов с широтно-импульсной модуляцией. Если записать лог. 1 в бит FOC0, то это приводит к принудительной установке результата сравнения на входе блока формирования выходного сигнала. Выход OC0 изменяется в соответствии с установками бит COM01, COM00. Следовательно, значение записанное в COM01, COM00 определяет эффект действия принудительной установки результата сравнения. Обратите также внимание, что бит FOC0 является стробирующим.

Строб FOC0 не генерирует каких-либо прерываний, а также не вызывает сброс таймера в режиме CTC, где регистр OCR0 задает верхний предел счета.

Бит FOC0 всегда считывается как 0.

Разряд 6, 3 - WGM01:0: Режим работы таймера-счетчика 0

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 52 представлены режимы работы таймера-счетчика 0 (см. также "Режимы работы таймера-счетчика 0").

Таблица 52. Описание бит, задающих режим работы таймера-счетчика 0

Номер режима	WGM01	WGM00	Наименование режима работы таймера-счетчика 0	Верхний предел счета	Условие обновления содержимого регистра OCR0	Условие установки флага TOV0
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	OCR0	Сразу после записи в регистр	Достижение максимального значения (0xFF)
3	1	1	Быстрая ШИМ	0xFF	Достижение верхнего предела счета	Достижение максимального значения (0xFF)

Разряд 5:4 - COM01, COM00: Режим формирования выходного сигнала

Данные биты определяют алгоритм изменения сигнала на выводе OC0. Если значение данных бит ненулевое, то функция вывода OC0 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 OC0 также необходимо установить бит 4 (OC0) в регистре DDRB для установки выходного направления.

После активизации альтернативной функции назначение бит COM01, COM00 зависит от выбранного режима работы таймера битами WGM01, WGM00. В таблице 53 приведено назначение бит COM01, COM00, если с помощью WGM01, WGM00 задан нормальный режим или режим сброса при совпадении (т.е. режимы без ШИМ).

Таблица 53. Режимы формирования выходного сигнала в режимах работы таймера 0 без ШИМ

COM01	COM00	Описание
0	0	Функция обычного порта ввода-вывода. OC0 отключен.
0	1	Переключение (инвертирование) OC0 при каждом совпадении
1	0	Сброс OC0 при каждом совпадении
1	1	Установка OC0 при каждом совпадении

В таблице 54 приведено назначение бит COM01, COM00 для режима работы таймера-счетчика 0 с быстрой ШИМ (WGM01:0).

Таблица 54. Режимы формирования выходного сигнала в режиме таймера 0 с быстрой ШИМ(1)

COM01	COM00	Описание
0	0	Функция обычного порта ввода-вывода. OC0 отключен.
0	1	Зарезервировано
1	0	Сброс OC0 при совпадении, установка по достижении верхнего предела (0xFF)
1	1	Установка OC0 при совпадении, сброс по достижении верхнего предела (0xFF)

Прим. 1: Имеется особый случай, когда OCR0 = 0xFF и COM01=1. В этом случае возникновение совпадения игнорируется, но сброс или установка по достижении верхнего предела выполняется. См. "Режим быстрой ШИМ".

В таблице 55 приведено действие бит COM01, COM00 для режима ШИМ с фазовой коррекцией, заданного с помощью бит WGM01, WGM00.

Таблица 54. Режимы формирования выходного сигнала в режиме ШИМ с фазовой коррекцией(1)

COM01	COM00	Описание
0	0	Функция обычного порта ввода-вывода. OC0 отключен.
0	1	Зарезервировано
1	0	Сброс OC0 при совпадении во время прямого счета. Установка OC0 при совпадении во время обратного счета.
1	1	Установка OC0 при совпадении во время прямого счета. Сброс OC0 при совпадении во время обратного счета.

Прим. 1: Существует особый случай, когда OCR0=0xFF/0x00 и COM01=1. В этом случае OC0 всегда находится на постоянном логическом уровне 0 или 1, т.к. 0xFF и 0x00 -точки изменения направления счета и возникающее на них совпадение зачитывается только к одному из направлений счета: обратному или прямому (см. также "Режим ШИМ с фазовой коррекцией").

Разряд 2:0 - CS02:0: Настройка частоты синхронизации таймера

С помощью трех настроечных бит имеется возможность выбрать различные тактовые частоты, кратные исходной частоте синхронизации (см. табл. 56).

Таблица 56. 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)

Регистр таймера-счетчика - TCNT0

Разряд	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр таймера-счетчика характеризуется двунаправленностью доступа к 8-разрядному счетчику таймера 0. Запись в регистр TCNT0 блокирует отработку возникающего совпадения на следующем после записи такте синхронизации таймера. Изменение содержимого счетчика (TCNT0) во время счета связано с риском потери результата сравнения между TCNT0 и регистром OCR0.

Регистр порога сравнения - OCR0

Разряд	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Регистр порога сравнения содержит 8-разр. значение, которое непрерывно сравнивается цифровым компаратором со значением 8-разр. счетчика (TCNT0). Факт совпадения значений может использоваться для генерации прерывания по выполнению условия сравнения или для генерации прямоугольных импульсов на выводе OC0.

Асинхронная работа таймера-счетчика 0

Регистр асинхронного состояния - ASSR

Разряд	7	6	5	4	3	2	1	0	
	-	-	-	-	AS0	TCN0UB	OCR0UB	TCR0UB	ASSR
Чтение/запись	Чт.	Чт	Чт	Чт	Чт./Зп.	Чт	Чт	Чт	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 3 - AS0: Разрешение асинхронного тактирования таймера-счетчика 0

Если AS0 = 0, то таймер счетчик 0 тактируется сигналом синхронизации ввода-вывода - clkI/O. Если AS0 = 1, то таймер-счетчик 0 тактируется низкочастотным кварцевым генератором, связанного с задающим кварцем через выводы TOSC1 и TOSC2. При изменении значения AS0 содержимое регистров TCNT0, OCR0 и TCCR0 может быть нарушено.

Разряд 2 - TCN0UB: Флаг занятости таймера-счетчика 0 при обновлении

Если таймер-счетчик 0 работает асинхронно и выполнена запись в TCNT0, то данный флаг устанавливается. После того, как содержимое TCNT0 обновляется из временного регистра, данный флаг сбрасывается аппаратно. Следовательно, когда TCN0UB=0, в TCNT0 может быть выполнена запись нового значения.

Разряд 1 - OCR0UB: Флаг занятости регистра порога сравнения при обновлении

Если таймер-счетчик 0 работает асинхронно и выполнена запись в регистр OCR0, то данный флаг устанавливается. По завершении обновления OCR0 из временного регистра данный флаг сбрасывается аппаратно. Если OCR0UB=0, то это означает готовность регистра OCR0 к записи нового значения.

Разряд 0 - TCR0UB: Флаг занятости регистра управления таймера-счетчика 0 при обновлении

Если таймер-счетчик 0 работает асинхронно и выполнена запись в регистр TCCR0, то данный флаг устанавливается. По завершении обновления TCCR0 из временного регистра данный флаг сбрасывается аппаратно. Если TCR0UB=0, то это означает готовность регистра TCCR0 к записи нового значения.

Если выполнить запись в любой из трех регистров таймера-счетчика 0, когда соответствующий флаг занятости установлен, то обновляемое значение может быть нарушено и может стать причиной несанкционированного возникновения прерывания.

Механизм чтения TCNT0, OCR0 и TCCR0 различный. Если выполняется чтение TCNT0, то считывается фактическое значение таймера. Если же считывается значение OCR0 или TCCR0, то фактически считывается содержимое временного регистра.

Асинхронная работа таймера-счетчика 0

Если таймер-счетчик 0 работает асинхронно, то необходимо учесть некоторые особенности.

Предупреждение: При переключении между асинхронным и синхронным тактовыми источниками таймера-счетчика содержимое регистров TCNT0, OCR0 и TCCR0 может быть нарушено. Во избежание этого необходимо придерживаться следующей безопасной последовательности переключения:

1. Отключить прерывания таймера-счетчика 0 путем сброса бит OCIE0 и TOIE0.
2. Выбрать необходимый тактовый источник с помощью бита AS0
3. Выполнить запись новых значений в TCNT0, OCR0 и TCCR0.
4. При переходе в асинхронный режим тактирования дождаться сброса флагов TCN0UB, OCR0UB и TCR0UB.
5. Сбросить флаги прерывания таймера-счетчика 0
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. о Запись в любой из регистров TCNT0, OCR0 или TCCR0 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 TCNT0 не влияет на процесс записи в регистр OCR0. Чтобы определить в какой регистр была выполнена запись, реализован регистр асинхронного состояния ASSR.
Если экономичный режим или расширенный дежурный режим вводится после записи в TCNT0, OCR0 или TCCR0, то программист должен дождаться завершения обновления записанного регистра, в случае если таймер-счетчик 0 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 0 используется для пробуждения микроконтроллера, т.к. функция отработки условия совпадения блокируется после записи в OCR0 или TCNT0. Если цикл записи не заканчивается и микроконтроллер переводится в режим сна прежде чем OCR0UB станет равным нулю, то микроконтроллер больше не будет прерываться при выполнении условия сравнения и, следовательно, не сможет пробудиться.
Если таймер-счетчик 0 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 TCCR0, TCNT0 или OCR0.
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 0 после подачи питания или выхода из режима выключения или дежурного режима. Содержимое всех регистров таймера-счетчика 0 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 0 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 TCNT0 сразу после пробуждения из экономичного режима (Power-save) может дать некорректный результат. Поскольку TCNT0 тактируется от асинхронного источника TOSC, то чтение TCNT0 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 TCNT0 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 TCNT0, рекомендуется соблюдать следующую последовательность действий:

1. По усмотрению записать произвольное значение или в регистр OCR0 или в TCCR0.
2. Дождаться сброса флага занятости при обновлении, соответствующего выбранному в п.1 регистру.
3. Выполнить чтение TCNT0.

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 OC0 привязан к синхронизации таймера и не синхронизирован с тактированием ЦПУ.

Регистр маски прерываний таймеров-счетчиков - TIMSK

Разряд	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 1 - OCIE0: Разрешение прерывания по результату сравнения таймера-счетчика 0

Если OCIE0=1, а также установлен бит I в регистре статуса, то прерывание по результату сравнения таймера-счетчика 0 активизируется. В этом случае прерывание возникает, если обнаруживается совпадение значения таймера-счетчика 0 с порогом сравнения, т.е. когда установлен флаг OCF0 в регистре флагов прерываний таймеров-счетчиков TIFR.

Разряд 0 - TOIE0: Разрешение прерывания по переполнению таймера-счетчика 0

Если TOIE0=1, а также установлен бит I в регистре статуса, то прерывание по переполнению таймера-счетчика 0 разрешается. В этом случае запрос на прерывание генерируется, если обнаруживается переполнение таймера-счетчика 0, т.е. когда установлен флаг TOV0 в регистре флагов прерываний таймеров-счетчиков TIFR.

Регистр флагов прерываний таймеров-счетчиков - TIFR

Разряд	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 1 - OCF0: Флаг совпадения таймера-счетчика 0

OCF0 равен лог. 1, если обнаруживается совпадением между значением таймера-счетчика 0 и данными в регистре OCR0 (регистр порога сравнения). OCF0 сбрасывается аппаратно при переходе на соответствующий вектор прерывания. Альтернативно, флаг OCF0 может быть сброшен путем записи в него лог. 1. Если установлены бит I в регистре SREG, бит OCIE0 (разрешено прерывание по выполнению условия сравнения таймера-счетчика 0) и флаг OCF0, то генерируется прерывание по выполнению условия сравнения таймера-счетчика 0.

Разряд 0 - TOV0: Флаг переполнения таймера-счетчика 0

Флаг TOV0 устанавливается, если в таймере-счетчике 0 возникает переполнение. Флаг TOV0 сбрасывается аппаратно при переходе на соответствующий вектор прерывания. Альтернативно, флаг TOV0 сбрасывается путем записи в него лог. 1. Если установлены бит I в регистре SREG, бит TOIE0 (разрешено прерывание по переполнению таймера-счетчика 0) и флаг TOV0, то генерируется прерывание по переполнению таймера-счетчика 0. В режиме ШИМ данный флаг устанавливается, если таймер-счетчик 0 изменяет направление счета на значении 0x00.

Предделитель таймера-счетчика 0

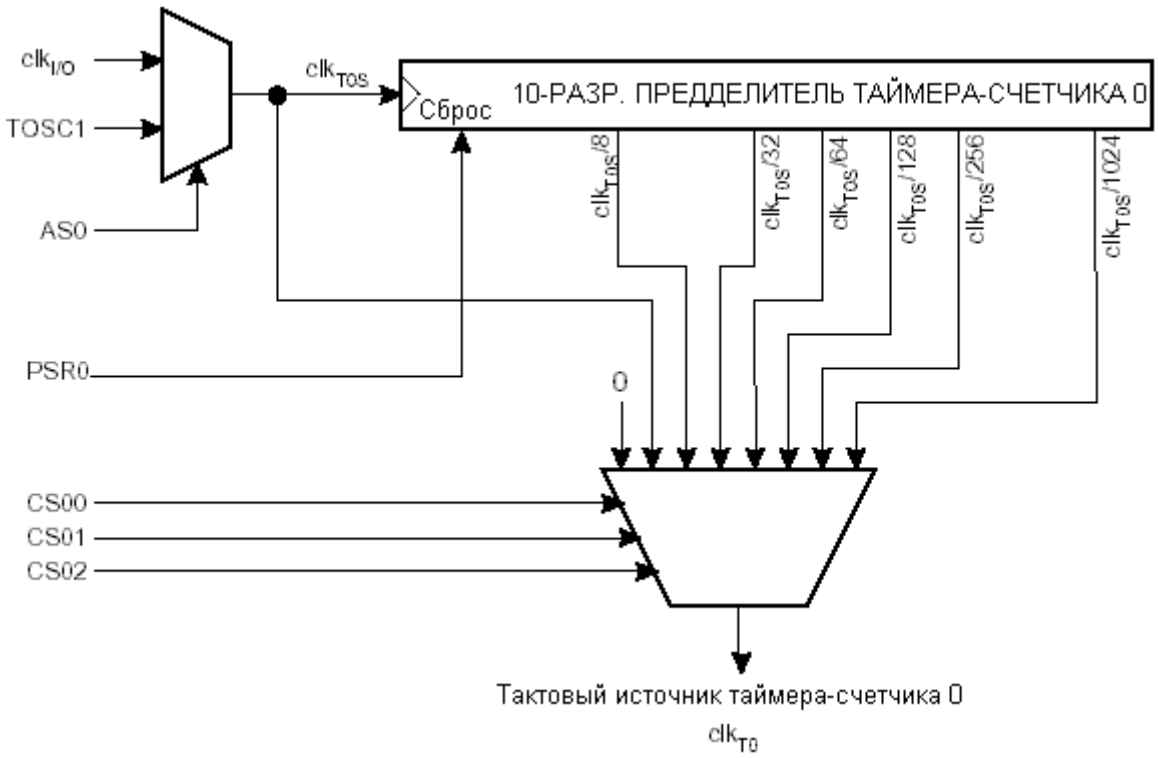


Рисунок 45. Предделитель таймер-счетчика 0

Тактовый источник таймера-счетчика 0 обозначен как clkT0. По умолчанию clkT0 подключен к системному источнику синхронизации ввода-вывода clkI/O. Путем установки бита AS0 в регистре ASSR таймер-счетчик 0 тактируется асинхронно с вывода TOSC1. Данная функция позволяет использовать таймер-счетчик 0 в качестве часов реального времени (RTC). Если AS=1, то выводы TOSC1 и TOSC2 более не выполняют функции линий порта C, а между ними может быть подключен кварцевый резонатор в качестве отдельного тактового источника таймера-счетчика 0. Генератор оптимизирован под использование кварца на частоту 32768 Гц. Подключение к выводу TOSC1 внешнего тактового источника не рекомендуется.

Предделитель таймера-счетчика 0 позволяет выбрать следующие тактовые сигналы: clkT0S/8, clkT0S/32, clkT0S/64, clkT0S/128, clkT0S/256 и clkT0S/1024. Кроме того, имеется возможность остановить синхронизацию. Установка бита PSR0 в регистре SFIOR сбрасывает предделитель. Данная функция позволяет программисту работать с более прогнозируемым поведением предделителя.

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.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 7 - TSM: Режим синхронизации таймера-счетчика

Установка бита TSM активизирует режим синхронизации таймеров-счетчиков. В данном режиме после установки бита PSR0 или PSR321 соответствующий предделитель будет постоянно находиться в сброшенном состоянии. В этом состоянии гарантируется, что все соответствующие таймеры-счетчики будут остановлены и могут быть одинаково настроены без риска изменения состояния одного из них во время конфигурации. Если после этого сбросить бит TSM, то биты PSR0 и PSR321 сбрасываются аппаратно, а таймеры-счетчики начинают счет одновременно.

Разряд 1 - PSR0: Сброс предделителя таймера-счетчика 0

Если данный бит равен лог. 1, то предделитель таймера-счетчика 0 сбрасывается. Данный бит обычно сбрасывается аппаратно сразу после установки. Если данный бит устанавливается, когда таймер-счетчик 0 работает в асинхронном режиме, то он остается равным 1 пока не сбросится предделитель таймера-счетчика 0. Данный бит не сбрасывается аппаратно, если бит TSM=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 детекцией 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

Два УСАПП

ATmega128 содержит два УСАПП: УСАПП0 и УСАПП1. Описание функционирования обоих УСАПП приведено ниже. УСАПП0 и УСАПП1 имеют раздельные регистры ввода-вывода, что показано в "Сводной таблице регистров". Обратите внимание, что в режиме совместимости с ATmega103 УСАПП1 не доступен, а также нет регистров UBRR0H и UCRS0C. Это означает, что в режиме совместимости с ATmega103 поддерживается только асинхронная работа УСАПП0.

Краткий обзор

На рисунке 79 представлена упрощенная функциональная схема УСАПП. На рисунке жирным шрифтом выделены регистры и выводы УСАПП.

Рисунок 79. Функциональная схема УСАПП

Прим. : Расположение выводов УСАПП см. на рисунке 1, табл. 76 и 39.

На рисунке 79 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 79) 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 80 показана функциональная схема логики синхронизации.

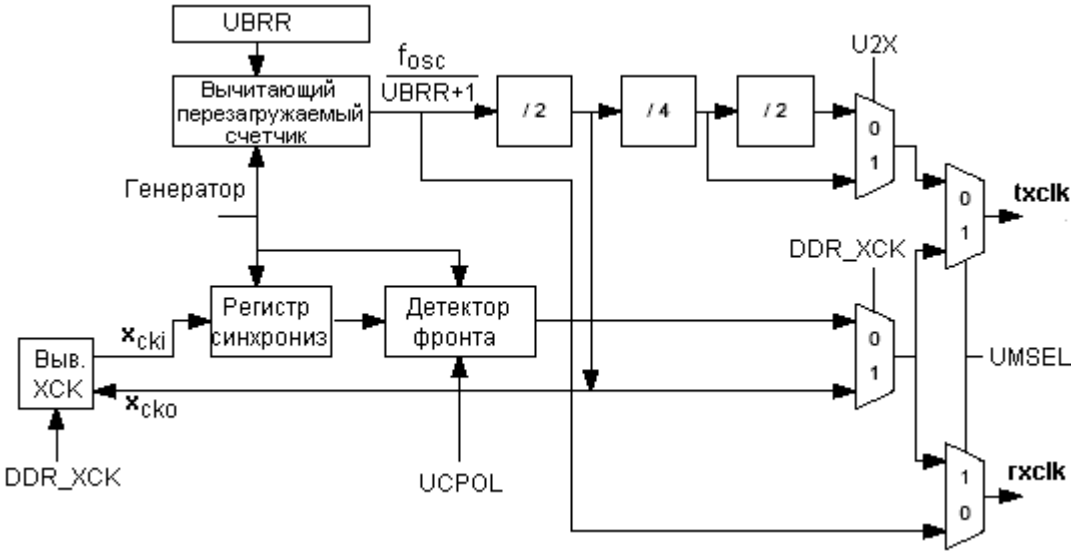


Рисунок 80. Функциональная схема логики синхронизации УСАПП

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. 80.

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 74 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 74. 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 82.

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. 80).

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).

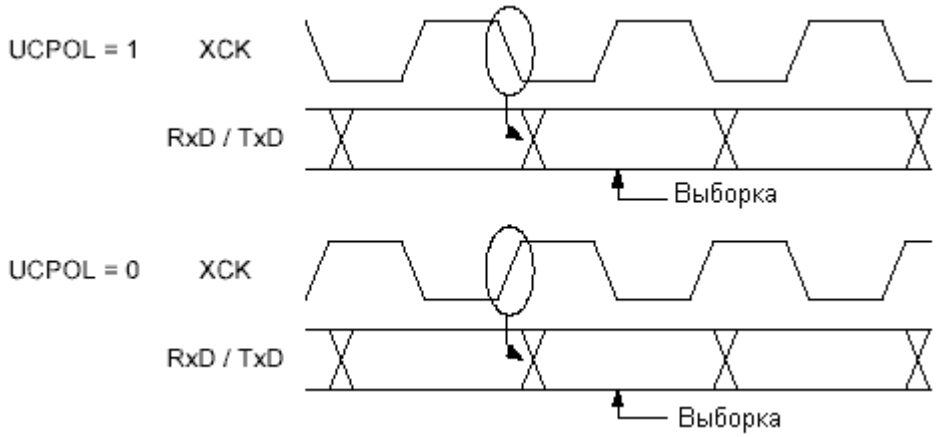


Рисунок 81. Временная диаграмма для синхронного режима XCK

Бит UCPOL регистра UCRSC выбирает какой фронт XCK используется для выборки данных, а какой для изменения данных. На рисунке 81 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 82 иллюстрирует возможность сочетания форматов посылки. Наличие прямоугольной скобки указывает на опциональность данного формата посылки.

Рисунок 82. Форматы посылки

- St - Старт-бит имеет всегда низкий уровень.
- 0…8 - Номер бита данных.
- P - бит паритета: четность или нечетность.
- Sp1,Sp2 - 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

$P_{ЧЕТН}$ - бит четного паритета;
$P_{НЕЧЕТН}$ - бит нечетного паритета;
d_n - 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.

Пример кода на Ассемблере [1]

```
USART_Init:
; Установка скорости связи
out UBRRH, r17
```

```
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<<USBS)|(3<<UCSZ0)
out UCSRC,r16
ret
```

Пример кода на Си [1]

```
void USART_Init( unsigned int baud )
{
/* Установка скорости связи */
UBRRH = (unsigned char)(baud>>8);
UBRRL = (unsigned char)baud;
/* Разрешение работы передатчика и приемника */
UCSRB = (1<<RXEN)|(1<<TXEN);
/* Установка формата посылки: 8 бит данных, 2 стоп-бита */
UCSRC = (1<<USBS)|(3<<UCSZ0);
}
```

Прим. 1: В примере предполагается, что подключен файл специфических заголовков. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода необходимо заменить инструкции "IN", "OUT", "SBIS", "SBIC", "CBI" и "SBI" на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции "LDS" и "STS" в сочетании с "SBRS", "SBRC", "SBR" и "CBR".

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
```

C Code Example [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: В примере предполагается, что подключен файл специфических заголовков. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода необходимо заменить инструкции "IN", "OUT", "SBIS", "SBIC", "CBI" и "SBI" на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции "LDS" и "STS" в сочетании с "SBRS", "SBRC", "SBR" и "CBR".

Перед загрузкой новых данных для передачи в данной функции осуществляется ожидание освобождения буфера передатчика путем опроса флага UDRE.

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. Для регистров ввода-вывода, которые расположены в области памяти

расширенного ввода-вывода, необходимо заменить инструкции "IN", "OUT", "SBIS", "SBIC", "CBI" и "SBI" на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции "LDS" и "STS" в сочетании с "SBRS", "SBRC", "SBR" и "CBR".

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: В примере предполагается, что подключен файл специфических заголовков. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода необходимо заменить инструкции "IN", "OUT", "SBIS", "SBIC", "CBI" и "SBI" на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции "LDS" и "STS" в сочетании с "SBRS", "SBRC", "SBR" и "CBR".

П

режде 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: В примере предполагается, что подключен файл специфических заголовков. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода необходимо заменить инструкции "IN", "OUT", "SBIS", "SBIC", "CBI" и "SBI" на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции "LDS" и "STS" в сочетании с "SBRS", "SBRC", "SBR" и "CBR".

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" и "Устройство проверки паритета"

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: В примере предполагается, что подключен файл специфических заголовков. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода, необходимо заменить инструкции "IN", "OUT", "SBIS", "SBIC", "CBI" и "SBI" на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции "LDS" и "STS" в сочетании с "SBRS", "SBRC", "SBR" и "CBR".

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. 83 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).

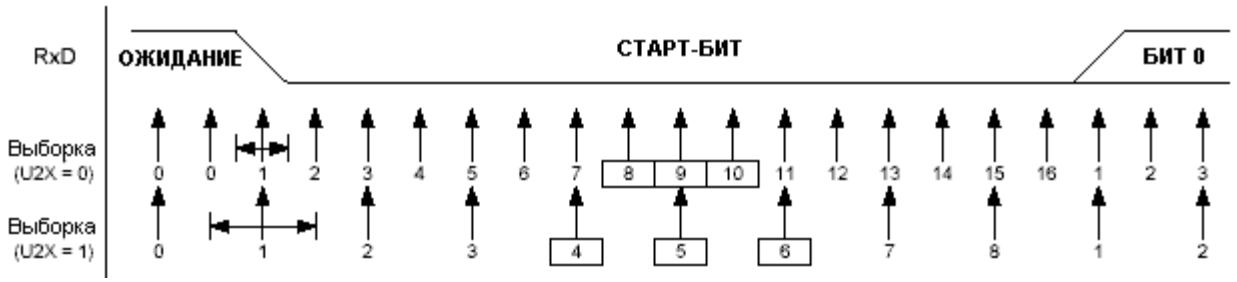


Рисунок 83. Выборка старт-бита

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 84 показана выборка бит данных и бита паритета. Для каждой выборки указан номер, который соответствует номеру состояния цифрового автомата блока обнаружения данных.

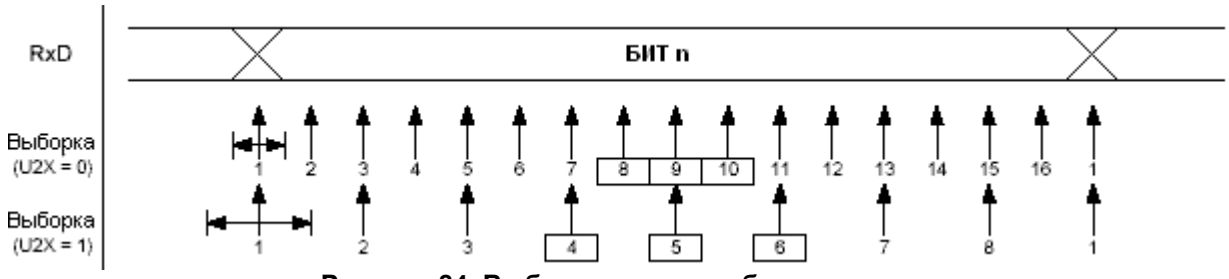


Рисунок 84. Выборка данных и бит паритета

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 85 отображен процесс выборки стоп-бита и начальный момент возможности обнаружения старт-бита следующей посылки.

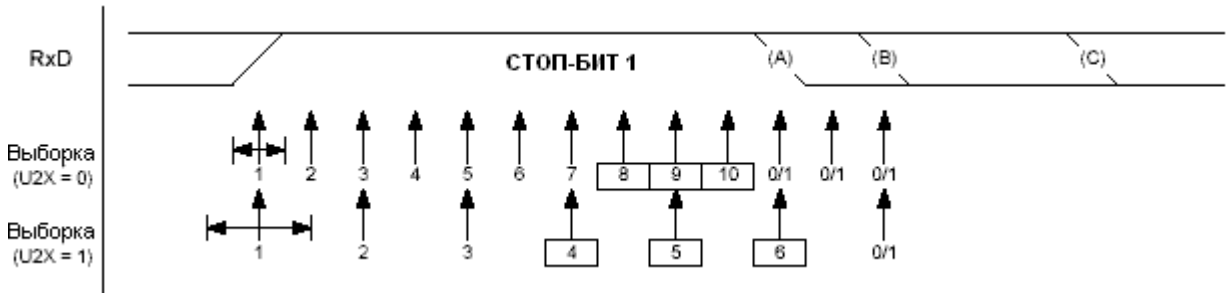


Рисунок 85. Выборка стоп-бита и следующего старт-бита

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. 75), 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:

$$R_{МИН} = \frac{(D+1)S}{S-1+D \cdot S + S_F}, R_{МАКС} = \frac{(D+2)S}{(D+1) \cdot S + S_M},$$

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 75 и 76 приведен список максимальных допустимых погрешностей при генерации скорости приемника. Обратите внимание, что режим нормальной скорости устойчив к более широким изменениям скорости связи.

Таблица 75. 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 76. 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.

Описание регистров УСАПП

Регистр данных УСАПП - UDRn

Разряд	7	6	5	4	3	2	1	0	
	RXBn[7:0]								UDRn (чтение)
	TXBn[7:0]								UDRn (запись)
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Буферные регистры данных передатчика и приемника УСАППn расположены по одному и тому же адресу в области ввода-вывода, обозначенной как регистр данных УСАППn или UDRn. Если выполнять запись по адресу регистра UDRn, то записываемые данные помещаются в буферный регистр данных передатчика TXBn. По аналогии, при чтении регистра UDRn извлекается содержимое буферного регистра данных приемника RXBn.

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 UDREn в регистре UCSRAn. Данные записанные в UDRn при сброшенном флаге UDREn будут игнорированы передатчиком УСАППn. 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 TxDn. 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 А управления и статуса УСАПП - UCSRnA

Разряд	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Чтение/запись	Чт.	Чт./Зп.	Чт.	Чт.	Чт.	Чт.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 7 - RXCn: 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 RXCn принимает нулевое значение. Флаг RXCn может использоваться для генерации прерывания по завершению приема (см. описание бита RXCIEn).

Разряд 6 - TXCn: Флаг завершения передачи УСАПП

Данный флаг устанавливается, если вся посылка из сдвигового регистра передатчика полностью передана и в передающем буфере UDRn нет новых данных для передачи. Флаг TXCn автоматически сбрасывается при переходе на вектор прерывания по завершению передачи или сбрасывается программно путем записи лог. 1 в позицию данного бита. Флаг TXCn может служить источником для генерации прерывания по завершению передачи (см. также описание бита TXCIEn).

Разряд 5 - UDREn: Флаг освобождения регистра данных УСАПП

Флаг UDREn индицирует о готовности приемного буфера UDRn к приему новых данных. Если UDREn=1, то буфер свободен и, следовательно, готов к записи. Флаг UDREn может служить источником для генерации прерывания по освобождению регистра данных (см. описание бит UDRIEn). UDREn устанавливается после сброса, индицируя о готовности передатчика.

Разряд 4 - FEn: 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 (UDRn). Флаг FEn принимает нулевое значение, если принятый стоп-бит имел правильное единичное значение. При записи в регистр UCSRnA в позиции данного бита необходимо указывать лог. 0.

Разряд 3 - DORn: 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 (UDRn). При записи в регистр UCSRnA в позиции данного бита необходимо указывать лог. 0.

Разряд 2 - UPEn: Ошибка паритета

Данный бит устанавливается, если следующая посылка в приемном буфере характеризуется ошибкой паритета, если во время приема этой посылки был разрешен контроль паритета (UPMn1 = 1). Данный бит имеет действительное значение до чтения приемного буфера (UDRn). При записи в регистр UCSRnA в позиции данного бита необходимо указывать лог. 0.

Разряд 1 - U2Xn: 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.

Разряд 0 - MPCMn: Режим многопроцессорной связи

Данный бит разрешает режим многопроцессорной связи. Если в бит MPCMn записать лог. 1, то все входящие посылки принимаемые приемником УСАПП будут игнорироваться, если они не содержат адресной информации. Установка бита MPCMn не влияет на работу передатчика. Более подробная информация по данному режиму приведена в параграфе "**Многопроцессорный режим связи**".

Регистр B управления и статуса УСАППn - UCSRnB

Разряд	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 7 - RXCIEn: Разрешение прерывания по завершению приема

Запись в данный бит лог. 1 разрешает прерывание по флагу RXCn. Прерывание по завершению приема УСАППn генерируется, если RXCIEn=1, флаг общего разрешения прерываний I=1 (в регистре SREG), а также установлен бит RXCn в регистре UCSRnA.

Разряд 6 - TXCIE: Разрешение прерывания по завершению передачи

Запись в данный бит лог. 1 разрешает прерывание по флагу TXCn. Прерывание по завершению передачи УСАППn генерируется, если TXCIEn=1, флаг общего разрешения прерываний I=1 (в регистре SREG), а также установлен бит TXCn в регистре UCSRnA.

Разряд 5 - UDRIEn: Разрешение прерывания по освобождению регистра данных УСАПП

Установка данного флага разрешает прерывание по флагу UDREn. Прерывание по освобождению регистра данных генерируется, если бит UDRIEn=1, флаг общего разрешения прерываний I=1 (в регистре SREG) и установлен бит UDREn в регистре UCSRnA.

Разряд 4 - RXENn: Разрешение работы приемника

Запись в данный бит лог. 1 приводит к разрешению работы приемника УСАППn. При этом, приемник формирует отключающий сигнал, который разрешает альтернативную функцию вывода RxDn. Отключение приемника приводит к сбросу приемного буфера, теряя при этом значения флагов FEn, DORn и UPEn.

Разряд 3 - TXENn: Разрешение работы передатчика

Запись в данный бит лог. 1 разрешает работу передатчика УСАППn. После этого передатчик генерирует отключающий сигнал, который активизирует альтернативную функцию вывода TxDn. Отключение передатчика (запись лог. 0 в TXENn) вступит в силу только по завершении генерации посылки, т.е. когда освободятся и сдвиговый регистр и буфер передатчика. После отключения вывод TxDn возвращается к выполнению функции обычной линии ввода-вывода.

Разряд 2 - UCSZn2: Формат данных

Бит UCSZn2 вместе с битами UCSZn1:0 в регистре UCSRnC задают количество бит данных в посылке, как для приемника, так и для передатчика.

Разряд 1 - RXB8n: Значение 8-ого разряда принятых данных

RXB8n содержит значение 9-го бита принятой посылки с 9-битным форматом. Данный бит необходимо считать прежде, чем будут считаны младшие 8 бит из регистра UDRn.

Разряд 0 - TXB8n: 8-ой разряд передаваемых данных

TXB8n 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 UDRn.

Регистр С управления и статуса УСАПП - UCSRnC

Разряд	7	6	5	4	3	2	1	0	
	-	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Обратите внимание, что данный регистр не доступен в режиме совместимости с ATmega103.

Разряд 7 - Резервный бит

Данный бит зарезервирован для будущего использования. Однако для совместимости с будущими микроконтроллерами при записи в регистр UCSRnC в позицию данного бита необходимо записывать лог. 0.

Разряд 6 - UMSELn: Выбор режима УСАПП

Данный бит позволяет переключаться между синхронным и асинхронными режимами последовательной связи.

Таблица 77. Установки бита UMSELn

UMSELn	Режим связи
0	Асинхронный
1	Синхронный

Разряды 5:4 - UPMn1:0: 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 UPEn в регистре UCSRnA.

Таблица 78. Установки бит UPMn

UPMn1	UPMn0	Режим паритета
0	0	Отключен
0	1	(Резерв)
1	0	Четность
1	1	Нечетность

Разряд 3 - USBSn: Выбор числа стоп-бит

Данный бит определяет сколько стоповых бит вставляет передатчик при генерации посылки. Приемник игнорирует эту настройку.

Таблица 79. Установки бита USBSn

USBSn	Число стоп-бит
0	1 бит
1	2 бита

Разряды 2:1 - UCSZn1:0: Формат данных

Биты UCSZn1:0 вместе с UCSZn2 в регистре UCSRnB задают количество бит данных в посылке, как для приемника, так и для передатчика.

Таблица 80. Установки бит UCSZn

UCSZn2	UCSZn1	UCSZn0	Формат данных
0	0	0	5 бит
0	0	1	6 бит
0	1	0	7 бит
0	1	1	8 бит
1	0	0	Резерв
1	0	1	Резерв
1	1	0	Резерв
1	1	1	9 бит

Разряд 0 - UCPOLn: 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 UCPOLn определяет соотношение между выборкой входящих данных и обновлением передаваемых данных и сигналом тактирования синхронной связи (XCKn).

Таблица 81. Установки бит UCPOLn

UCPOLn	Изменение передаваемых данных на выходе TxDn	Выборка принимаемых данных на входе RxDn
0	Нарастающий фронт XCKn	Падающий фронт XCKn
1	Падающий фронт XCKn	Нарастающий фронт XCKn

Регистры скорости связи УСАПП - UBRRnL и UBRRnH

Разряд	7	6	5	4	3	2	1	0	
	-	-	-	-	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
Чтение/запись	Чт.	Чт.	Чт.	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Регистр UBRRnH не доступен в режиме совместимости с mega103

Разряды 15:12 -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 UBRRnH.

Разряды 11:0 - UBRRn11:0: Регистр скорости связи УСАПП

UBRR - 12-разр. регистр, который задает значение скорости связи УСАПП. Регистр UBRRnH содержит 4 старших разряда, а UBRRnL 8 младших разрядов значения скорости УСАППn. Если во время передачи или приема изменить скорость связи, то сеанс связи будет нарушен. Запись в регистр UBRRnL инициирует обновление предделителя скорости связи.

Примеры установок скоростей связи

В таблице 82 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:

$$\delta = \left(\frac{fген}{16 \cdot (UBRR + 1) \cdot fсвязи} - 1 \right) \cdot 100,\% ,$$

где

$f_{ген}$ - частота тактового генератора, Гц;
$f_{связи}$ - скорость связи, бит/с;
UBRR - значение регистра UBRR.

Таблица 82. Примеры установок UBRR для типичных частот тактового генератора

Скорость связи, бит/с	$f_{ген}$ = 1.0000МГц				$f_{ген}$ = 1.8432 МГц				$f_{ген}$ = 2.0000 МГц			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	δ	UBRR	δ	UBRR	δ	UBRR	δ	UBRR	δ	UBRR	δ
2400 4800 9600 14.4k 19.2k 28.8k 38.4k	25 12 6 3 2 1 1	0.2% 0.2% -7.0% 8.5% 8.5% 8.5% - 18.6%	51 25 12 8 6 3 2	0.2% 0.2% 0.2% -3.5% -7.0% 8.5% 8.5%	47 23 11 7 5 3 2	0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%	95 47 23 15 11 7 5	0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%	51 25 12 8 6 3 2	0.2% 0.2% 0.2% -3.5% -7.0% 8.5% 8.5%	103 51 25 16 12 8 6	0.2% 0.2% 0.2% 2.1% 0.2% - 3.5% - 7.0%
valign="top"57.6k 76.8k 115.2k 230.4k 250k	0 - - - -	8.5% - - - -	1 1 0 - -	8.5% - 18.6% 8.5% - -	1 1 0 - -	0.0% - 25.0% 0.0% - -	3 2 1 0 -	0.0% 0.0% 0.0% 0.0%	1 1 0 - -	8.5% - 18.6% 8.5% - -	3 2 1 - 0	8.5% 8.5% 8.5% - 0.0%
Макс. (1)	62.5 кбит/с		125 кбит/с		115.2 кбит/с		230.4 кбит/с		125 кбит/с		250 кбит/с	

1. UBRR = 0, Погрешность = 0.0%

Таблица 83. Примеры установок UBRR для типичных частот тактового генератора

Скорость связи, бит/с	$f_{ген}$ = 3.6864 МГц				$f_{ген}$ = 4.0000 МГц				$f_{ген}$ = 7.3728 МГц			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	δ	UBRR	δ	UBRR	δ	UBRR	δ	UBRR	δ	UBRR	δ
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 1M	- -	- -	0 -	-7.8% -	- -	- -	0 -	0.0% -	0 -	-7.8% -	1 0	-7.8% -7.8%
Макс. (1)	230.4 кбит/с		460.8 кбит/с		250 кбит/с		0.5 Мбит/с		460.8 кбит/с		921.6 кбит/с	

1. UBRR = 0, Погрешность = 0.0%

Таблица 84. Примеры установок UBRR для типичных частот тактового генератора

Скорость связи, бит/с	$f_{ген}$ = 8.0000 МГц				$f_{ген}$ = 11.0592 МГц				$f_{ген}$ = 14.7456 МГц			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	δ	UBRR	δ	UBRR	δ	UBRR	δ	UBRR	δ	UBRR	δ
2400	207	0.2%	416	-	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.1%	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	-	68	0.2%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.8%	51	0.6%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	0.2%	34	0.2%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	2.1%	25	-	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	0.2%	16	0.8%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	3.5%	8	2.1%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	-	3	0.2%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	7.0%	3	-	2	-	5	-7.8%	3	-	6	5.3%
0.5M	0	8.5%	1	3.5%	-	7.8%	2	-7.8%	1	7.8%	3	-7.8%
1M	-	8.5% 0.0% 0.0% -	0	8.5% 0.0% 0.0% 0.0%	-	- -	-	-	0	- 7.8% - 7.8%	1	-7.8%
Макс. (1)	0.5 Мбит/с		1 Мбит/с		691.2 кбит/с		1.3824 Мбит/с		921.6 кбит/с		1.8432 Мбит/с	

1. UBRR = 0, Погрешность = 0.0%

Таблица 85. Примеры установок UBRR для типичных частот тактового генератора

Скорость связи, бит/с	$f_{ген}$ = 16.0000 МГц				$f_{ген}$ = 18.4320 МГц				$f_{ген}$ = 20.0000 МГц			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	δ	UBRR	δ	UBRR	δ	UBRR	δ	UBRR	δ	UBRR	δ
2400	416	-	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.1%	416	-	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.1%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.2%	138	0.2%	79	0.0%	159	0.0%	86	-	173	-
19.2k	51	0.6%	103	-	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	0.2%	68	0.1%	39	0.0%	79	0.0%	42	0.2%	86	0.2%
38.4k	25	-	51	0.2%	29	0.0%	59	0.0%	32	0.9%	64	-
57.6k	16	0.8%	34	0.6%	19	0.0%	39	0.0%	21	-	42	0.2%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.4%	32	0.2%
115.2k	8	2.1%	16	-	9	0.0%	19	0.0%	10	-	21	0.9%
230.4k	3	0.2%	8	0.8%	4	0.0%	9	0.0%	4	1.4%	10	-
250k	3	- 3.5% 8.5% 0.0%	7	0.2% 2.1% - 3.5% 0.0%	4	-7.8%	8	2.4%	4	1.7% - 1.4% 8.5% 0.0%	9	1.4% - 1.4% - 1.4% 0.0%
0.5M	1	0.0%	3	0.0%	-	-	4	-7.8%	-	-	4	0.0%
1M	0	0.0%	1	0.0%	-	-	-	-	-	-	-	-
Макс. (1)	1 Мбит/с		2 Мбит/с		1.152		2.304		1.25 Мбит/с		2.5 Мбит/с	

			Мбит/с	Мбит/с		

1. UBRR = 0, Погрешность = 0.0%

Двухпроводной 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.

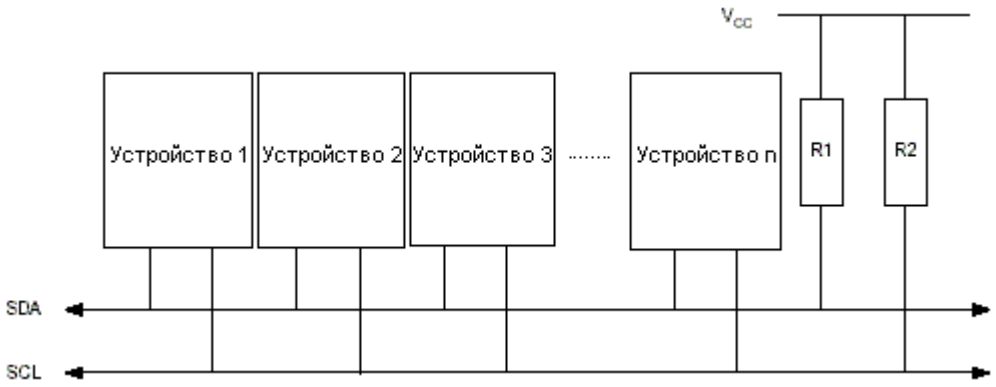


Рисунок 86. Внешние подключения к шине TWI

Терминология TWI

Ниже приведены часто используемые в данном разделе термины.

Таблица 86. 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.

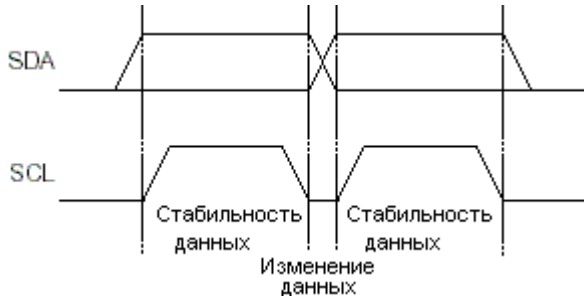


Рисунок 87. Действительность данных

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.

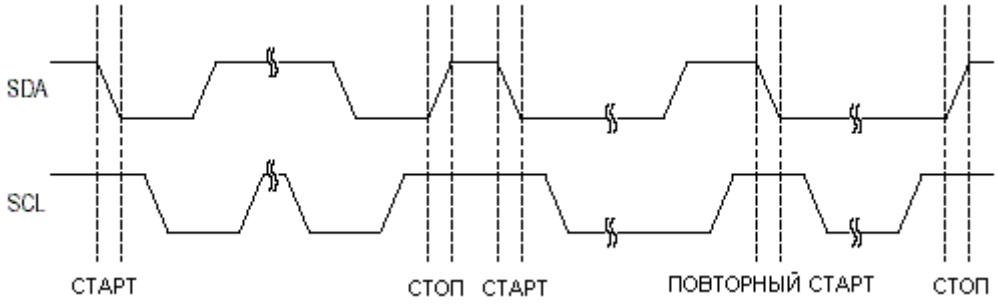


Рисунок 88. Условия СТАРТа, ПОВТОРНОГО СТАРТА и ОСТАНОВа

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.

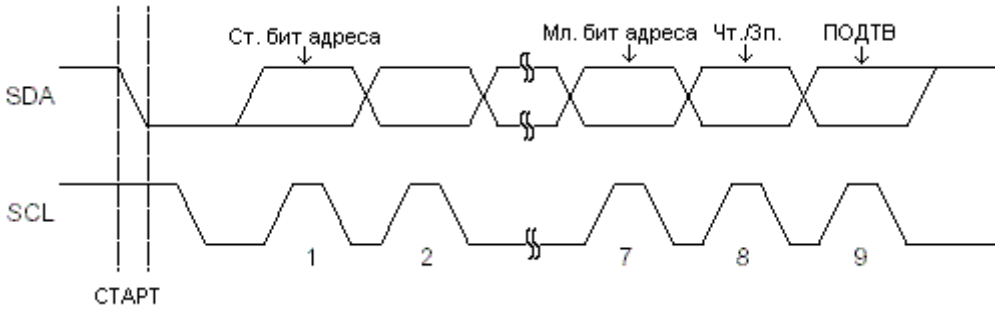


Рисунок 89. Формат адресного пакета

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.

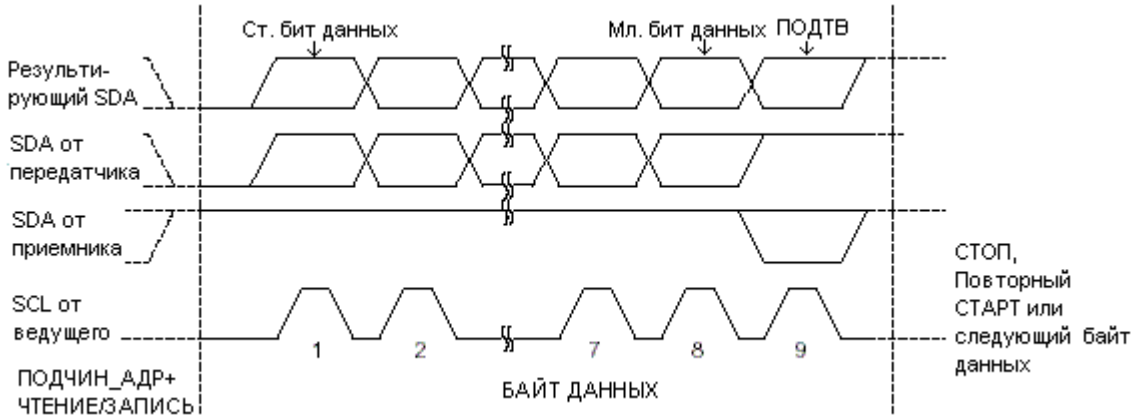


Рисунок 90. Формат пакета данных

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 91 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.

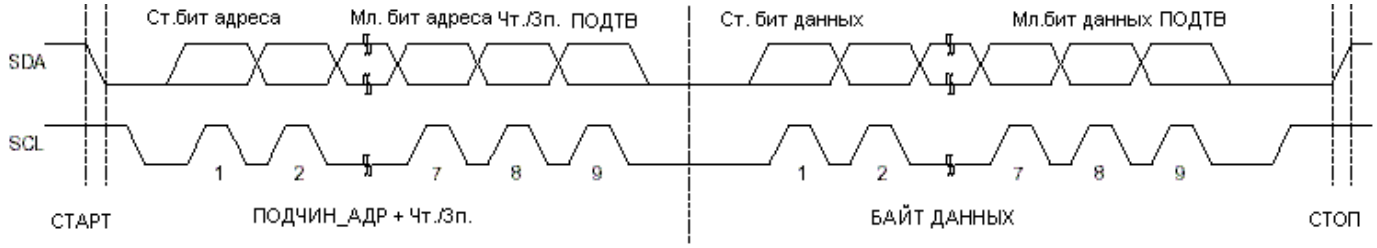


Рисунок 91. Типичная передача данных

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.

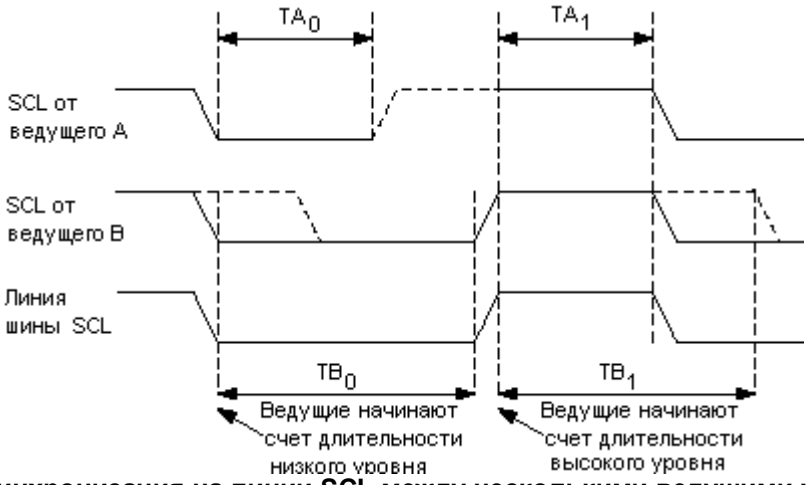


Рисунок 92. Синхронизация на линии SCL между несколькими ведущими устройствами

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.

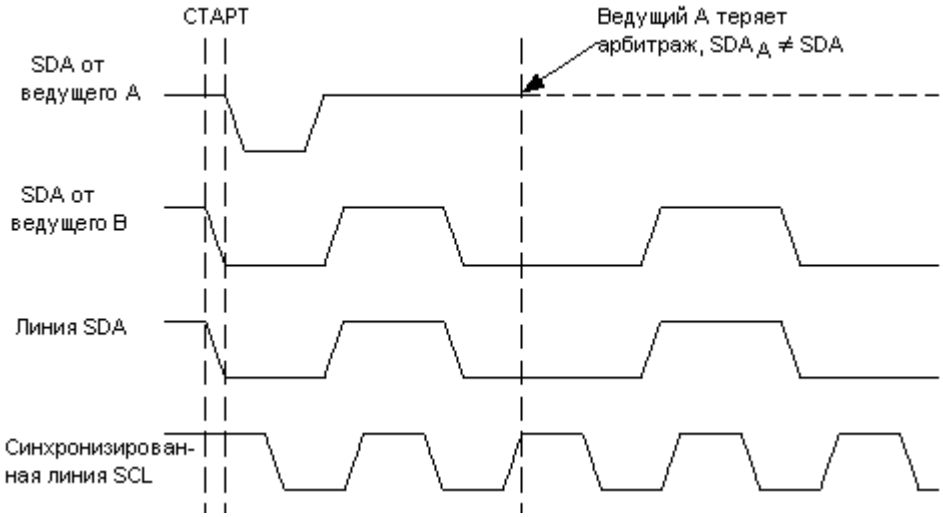


Рисунок 93. Арбитрирование двух мастеров

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 94). Все регистры выделенные жирной линией доступны через шину данных микроконтроллера.

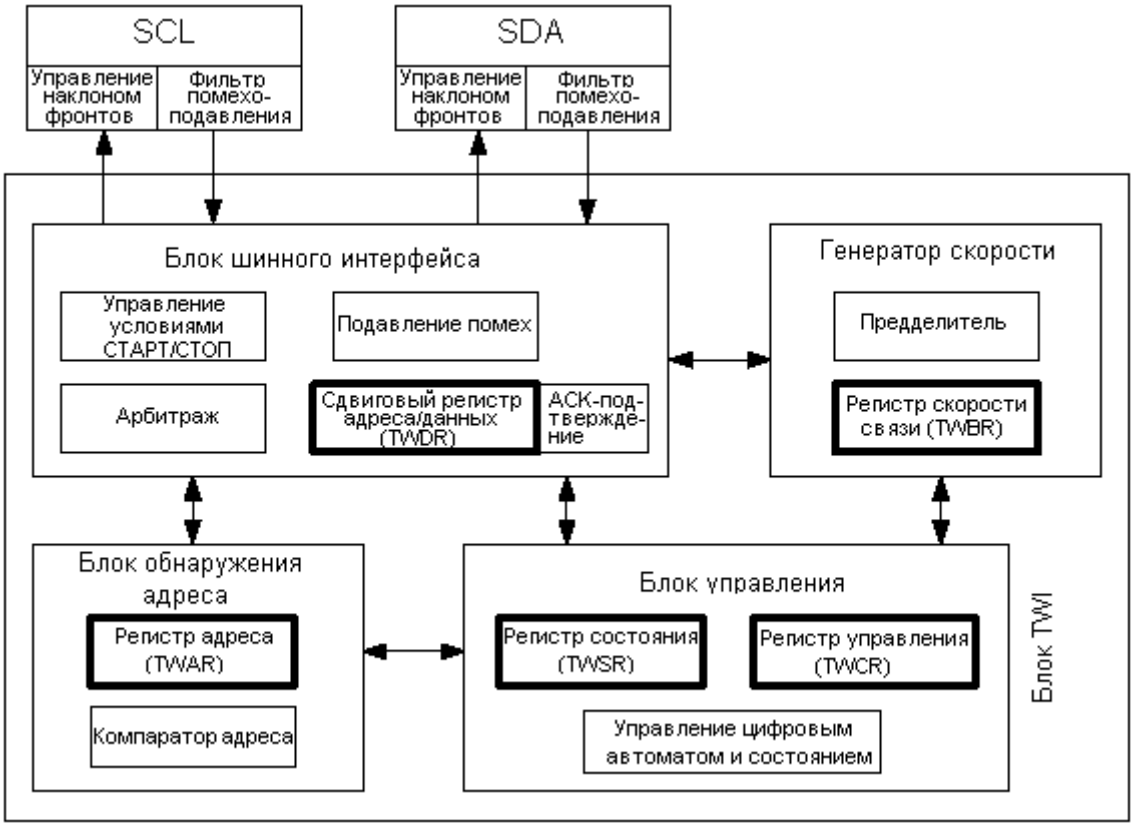


Рисунок 94. Функциональная схема модуля TWI

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.: TWBR должен быть равен не менее 10, если TWI работает в ведущем режиме. Если TWBR меньше 10, то ведущий может генерировать некорректное состояние на линиях SDA и SCL. Проблема возникает при работе в ведущем режиме при передаче условий СТАРТ+ПОДЧИН_АДР+ ЧТЕНИЕ/ЗАПИСЬ подчиненному.

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

Разряд	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	0	0	0	0	0	0	0	0	

Разряд 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.

Регистр управления шиной TWI - TWCR

Разряд	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	TWCR
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт.	Чт./Зп.	Чт.	Чт./Зп.	
Исх. значение	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.

Разряд 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.

Разряд 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.

Разряд 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.

Разряд 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.

Разряд 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.

Разряд 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.

Разряд 1 - Резервный бит

Данный бит является резервным и считывается как 0.

Разряд 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

Разряд	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0	TWSR
Чтение/запись	Чт.	Чт.	Чт.	Чт.	Чт.	Чт.	Чт./Зп.	Чт./Зп.	
Исх. значение	1	1	1	1	1	0	0	0	

Разряды 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.

Разряд 2 - Резервный бит

Данный бит является резервным и считывается как 0.

Разряды 1..0 - TWPS: Биты предделителя TWI

Данные биты отличаются полным доступом (чтение/запись) и позволяют управлять предделителем скорости связи.

Таблица 87. 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

Разряд	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWD
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	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.

Разряды 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

Разряд	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWD
Чтение/запись	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. значение	1	1	1	1	1	1	1	1	

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.

Разряды 7..1 - TWA: Регистр подчиненного адреса TWI

Данные семь бит составляют подчиненный адрес блока TWI.

Разряд 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 95 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.

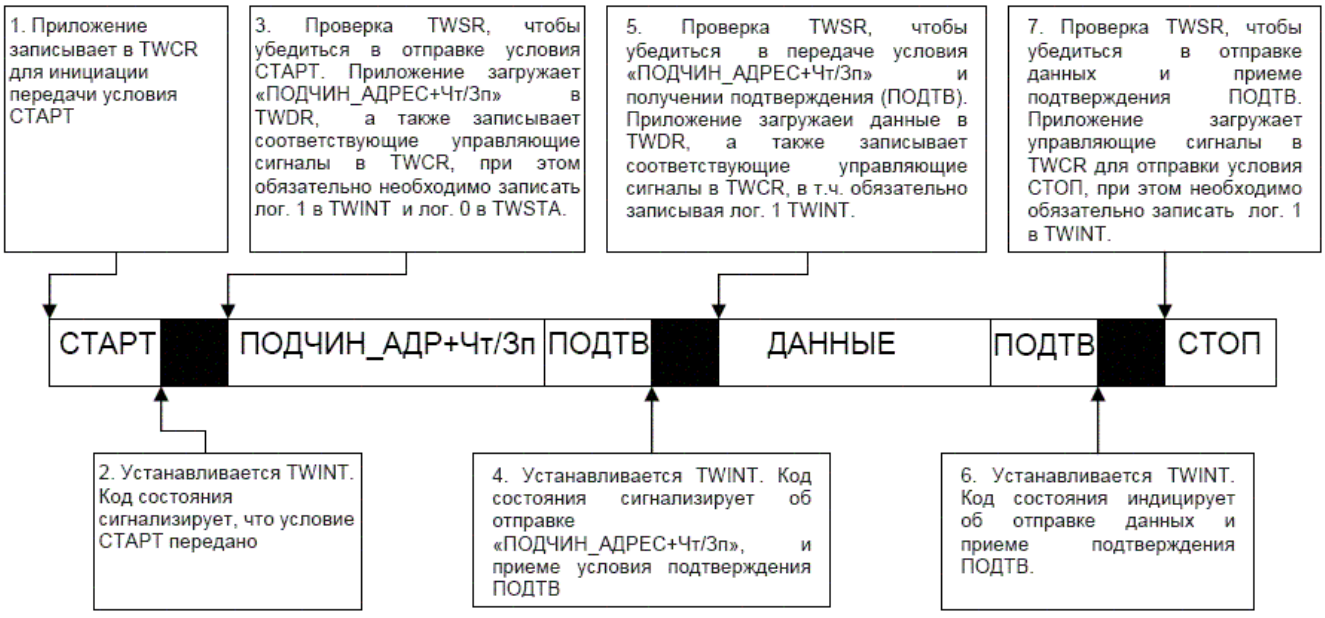


Рисунок 95. Последовательность обслуживания TWI при типичной передаче

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 sbrs r16,TWINT 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
	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

Прим.: Если фактический регистр расположен в расширенной памяти ввода-вывода, то инструкции "IN", "OUT", "SBIS", "SBIC", "CBI" и "SBI" необходимо заменить теми инструкциями,

которые эффективны для данной области памяти. Обычно это инструкции "LDS" и "STS" в сочетании с "SBRS", "SBRC", "SBR" и "CBR".

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 ЭСППЗУ, а режим MR для считывания данных из ЭСППЗУ. 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 97-103 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 96). 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.

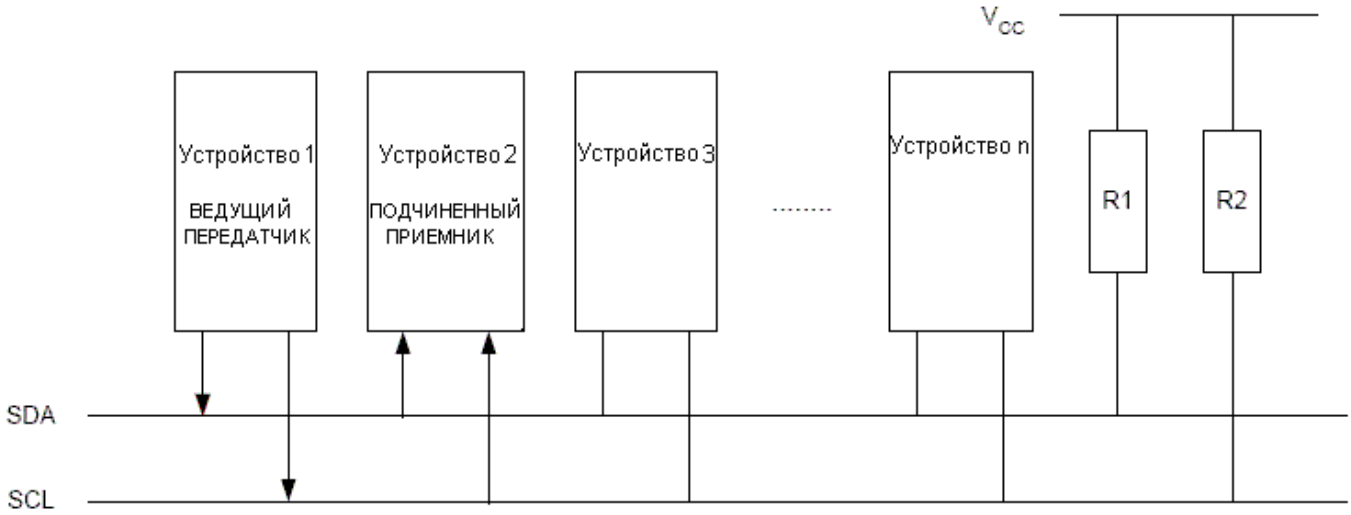


Рисунок 96. Передача данных в режиме ведущего передатчика

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 88.

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 88. Коды состояния в режиме ведущего передатчика

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	
$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

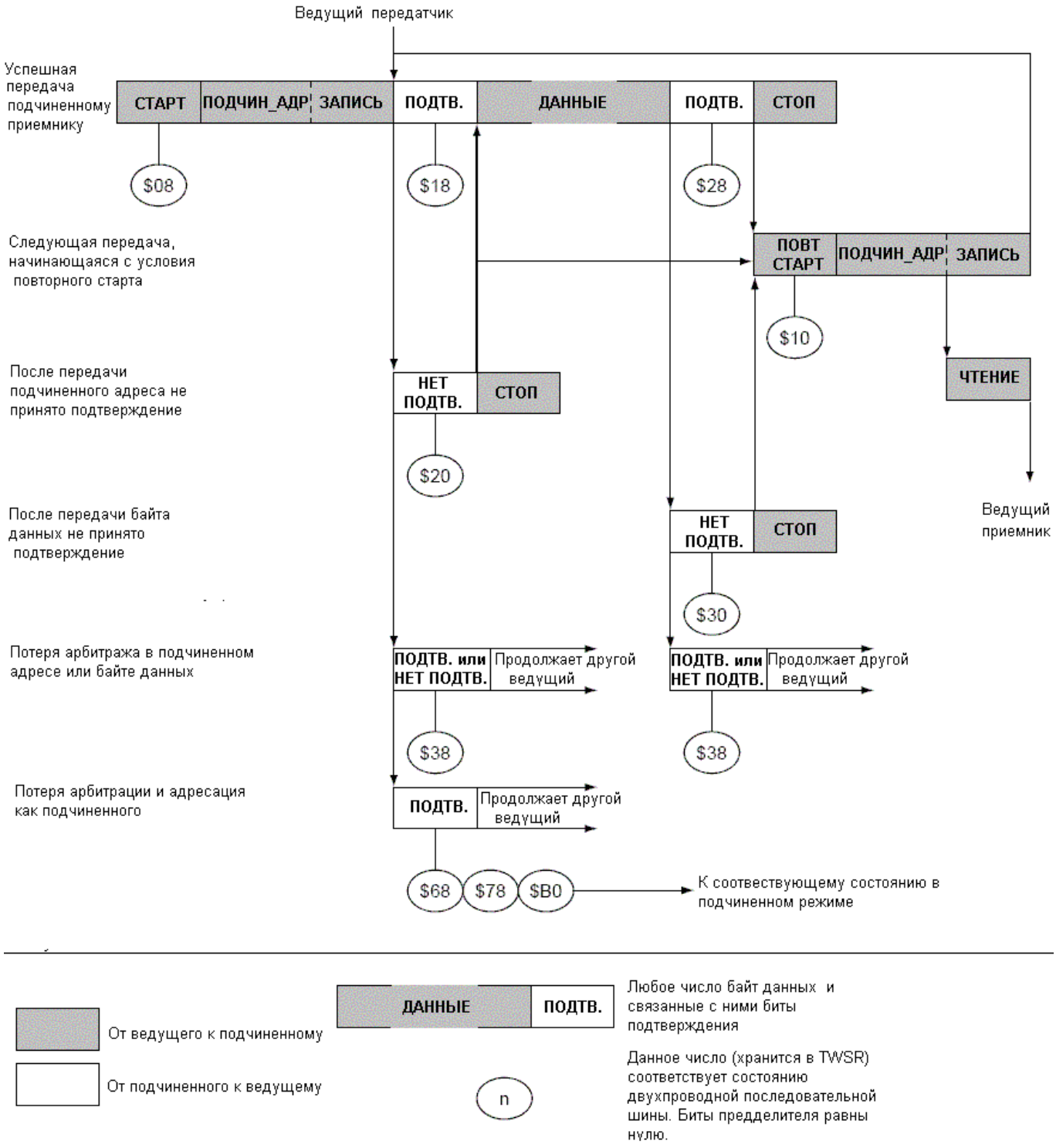


Рисунок 97. Форматы и состояния в режиме ведущего передатчика

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.

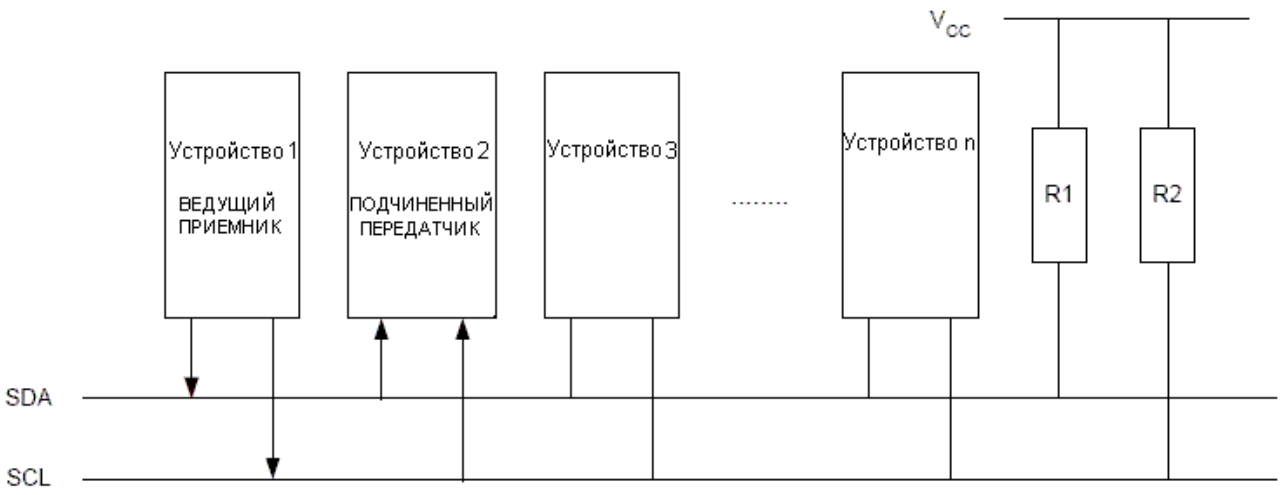


Рисунок 98. Передача данных в режиме ведущего приемника

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 88). 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 97. 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.

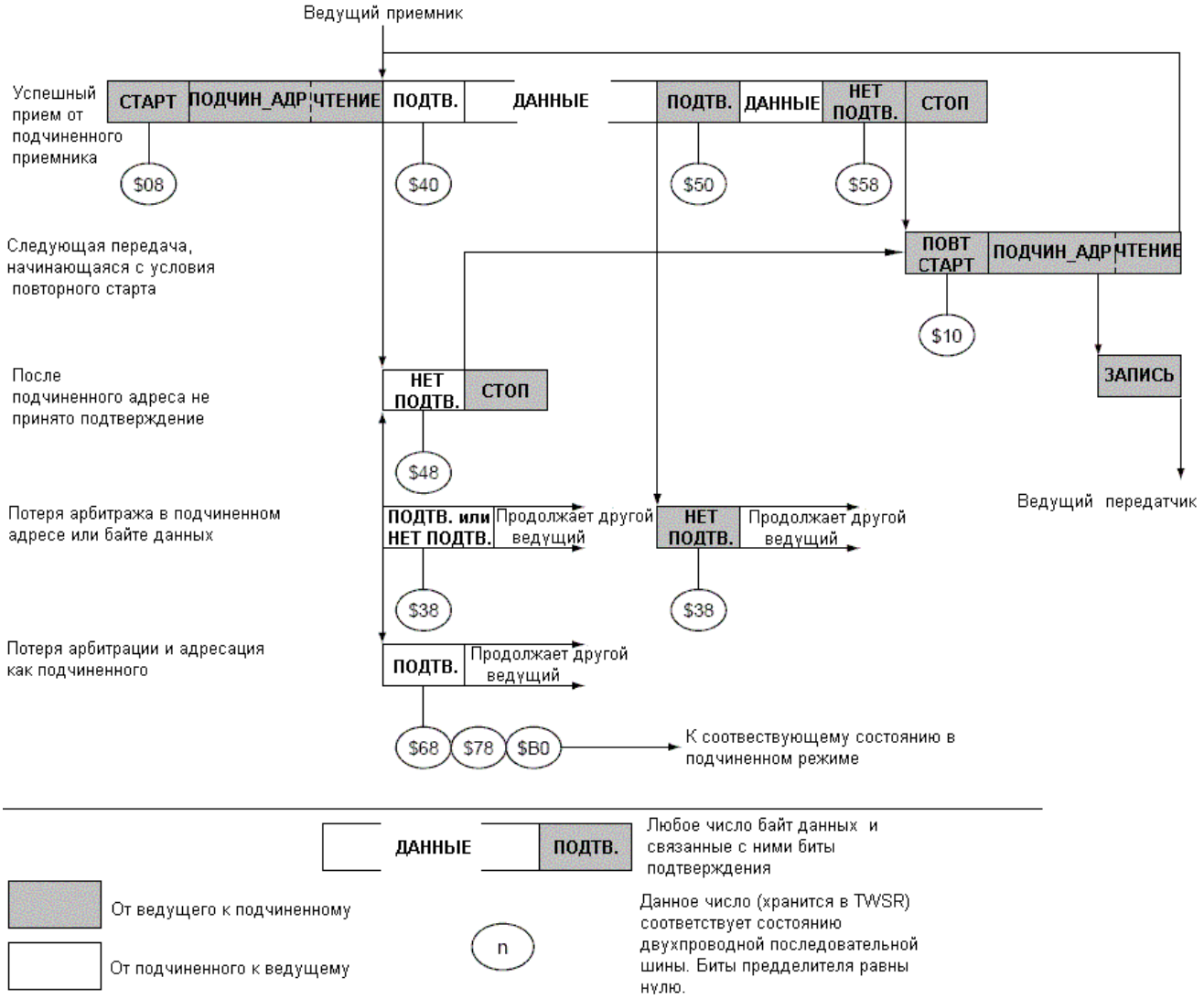


Рисунок 99. Форматы и состояния в режиме ведущего приемника

Таблица 89. 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
		В/из 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

Режим подчиненного приемника

В режиме подчиненного приемника принимается несколько байт данных от ведущего передатчика (см. рисунок 100). Во всех кодах состояния, приведенных в этом разделе, не учитываются биты предделителя и они равны нулю.

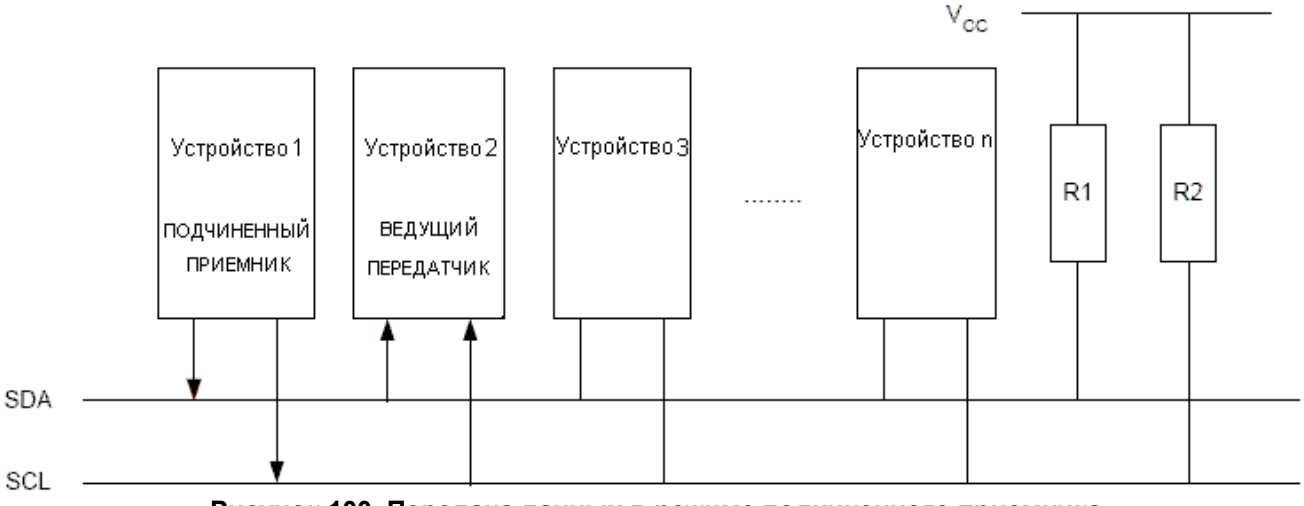


Рисунок 100. Передача данных в режиме подчиненного приемника

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 90. 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
		В/из TWDR	В TWCR				
			STA	STO	TWINT	TWEA	
$60	Принимается собственный ПОДЧИН_АДР+ЗАПИСЬ; возращено 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, возращено 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;
		чтение байта данных или					не распознается собственный
		чтение байта данных или					ПОДЧИН_АДР или адрес общего вызова
		чтение байта данных	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
$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 чтение байта данных или чтение байта данных или чтение байта данных	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
$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 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

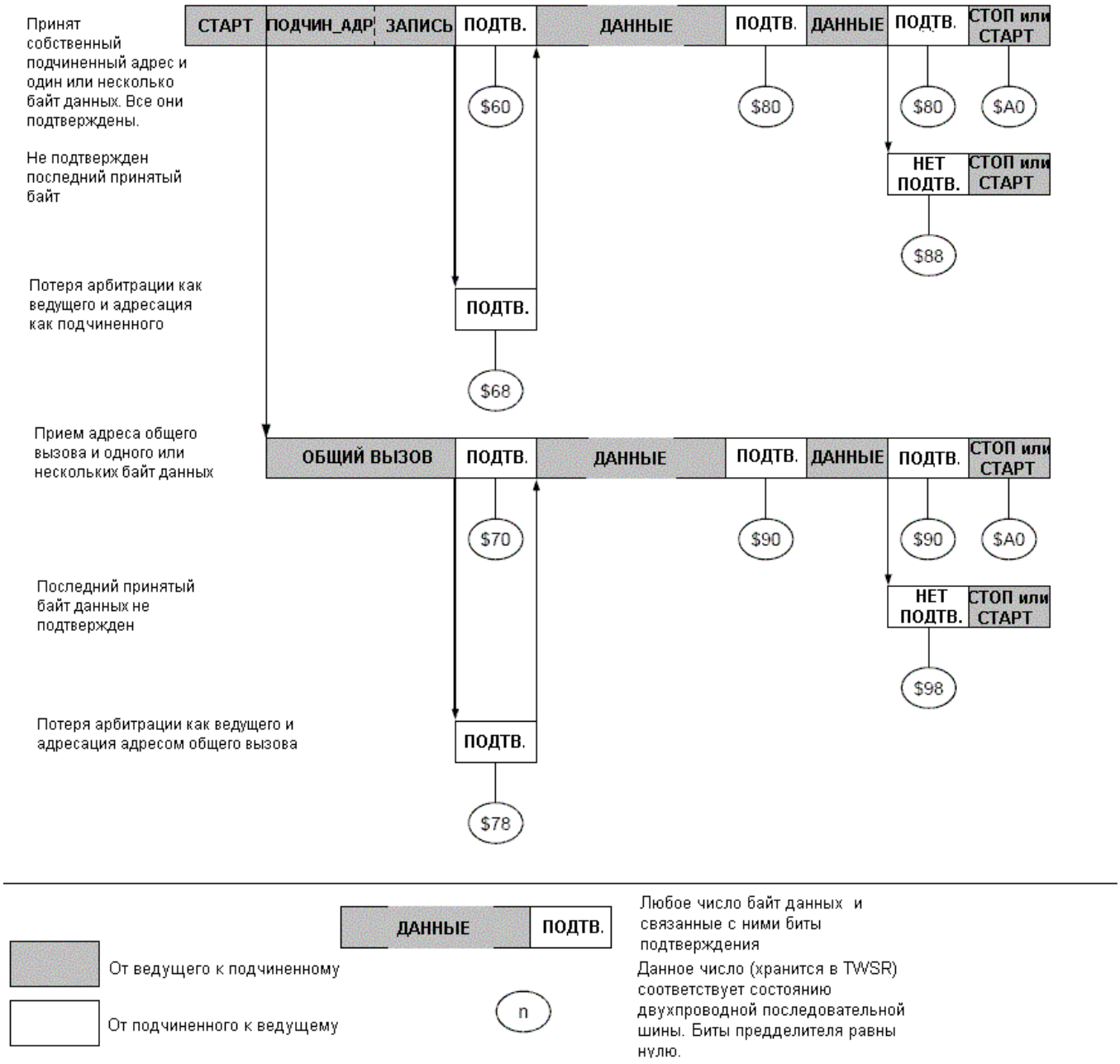


Рисунок 101. Форматы и состояния в режиме подчиненного приемника

Режим подчиненного передатчика

В режиме подчиненного передатчика выполняется передача нескольких байт данных ведущему приемнику (см. рисунок 102). Во всех кодах состояния, приведенных в этом разделе, не учитываются биты предделителя и они равны нулю.

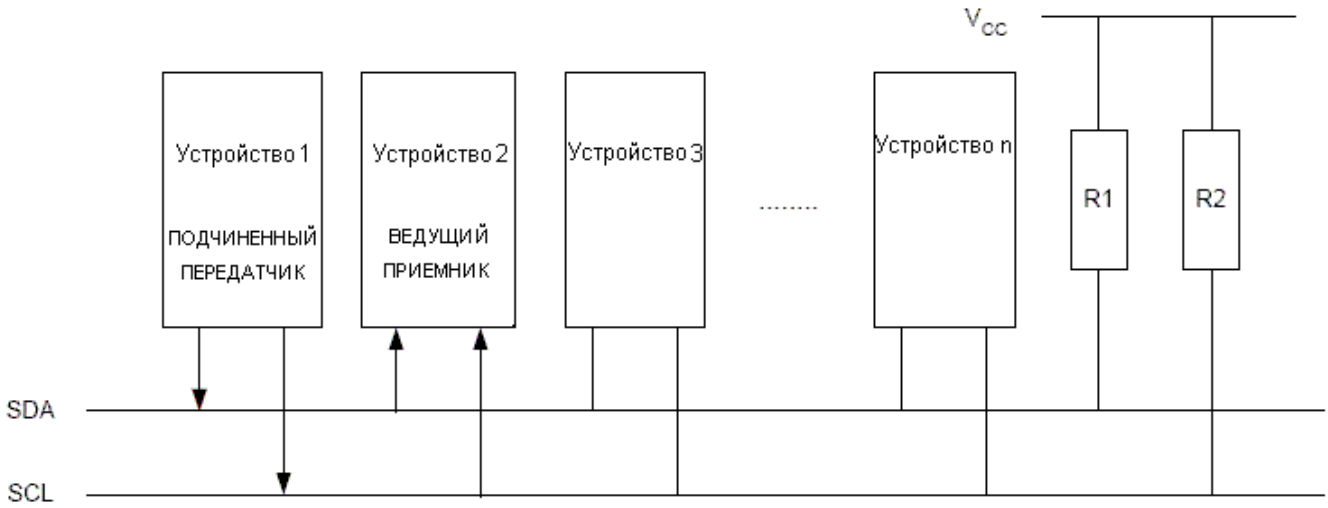


Рисунок 102. Передача данных в режиме подчиненного передатчика

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 91. 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 91. 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; возращено 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

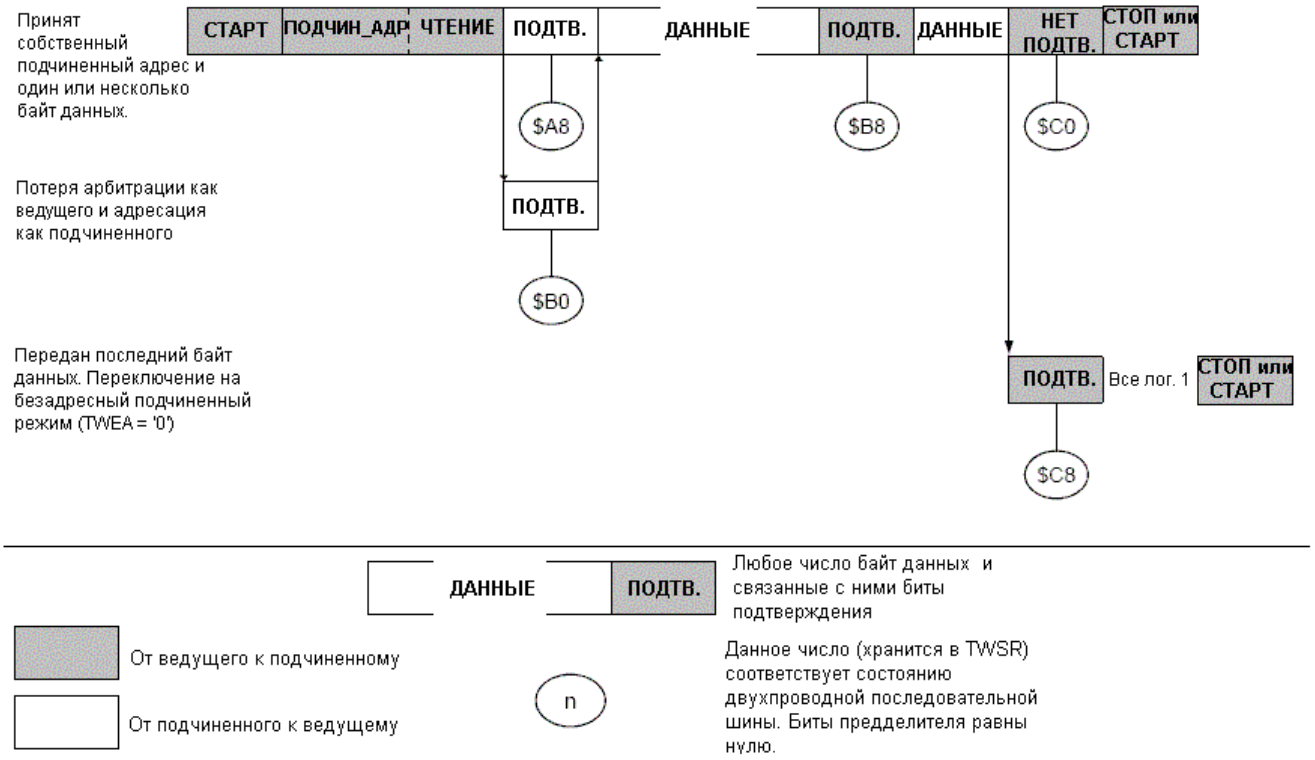


Рисунок 103. Форматы и состояния в режиме подчиненного передатчика

Прочие состояния

Имеются несколько кодов состояний, которые отличаются от упомянутых выше (см. табл. 92). 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 92. Прочие состояния

Сочетание нескольких режимов

В некоторых случаях сочетаются несколько режимов TWI для обеспечения желаемого действия. В качестве примера рассмотрим чтение данных из последовательного ЭСППЗУ. Обычно, такой сеанс связи организовывается в такой последовательности:

1. Инициируется сеанс связи
2. В ЭССПЗУ 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 ЭСППЗУ 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.

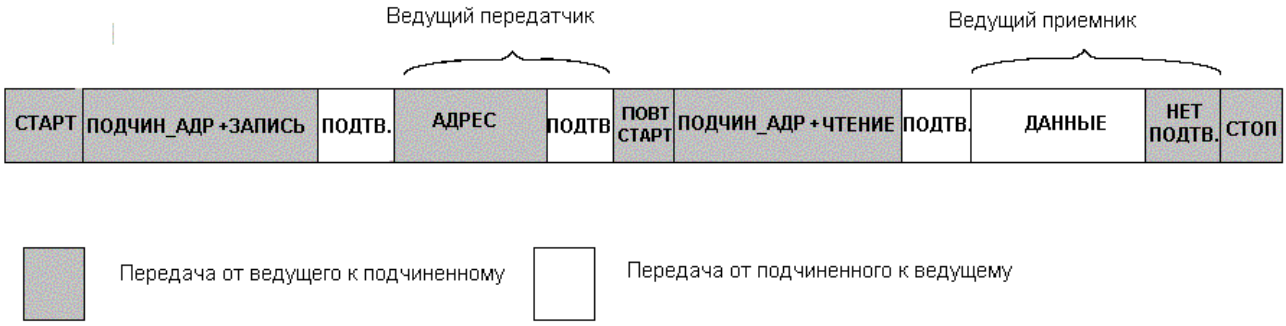


Рисунок 104. Сочетание нескольких режимов TWI для обмена данными с последовательным ЭСППЗУ

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.

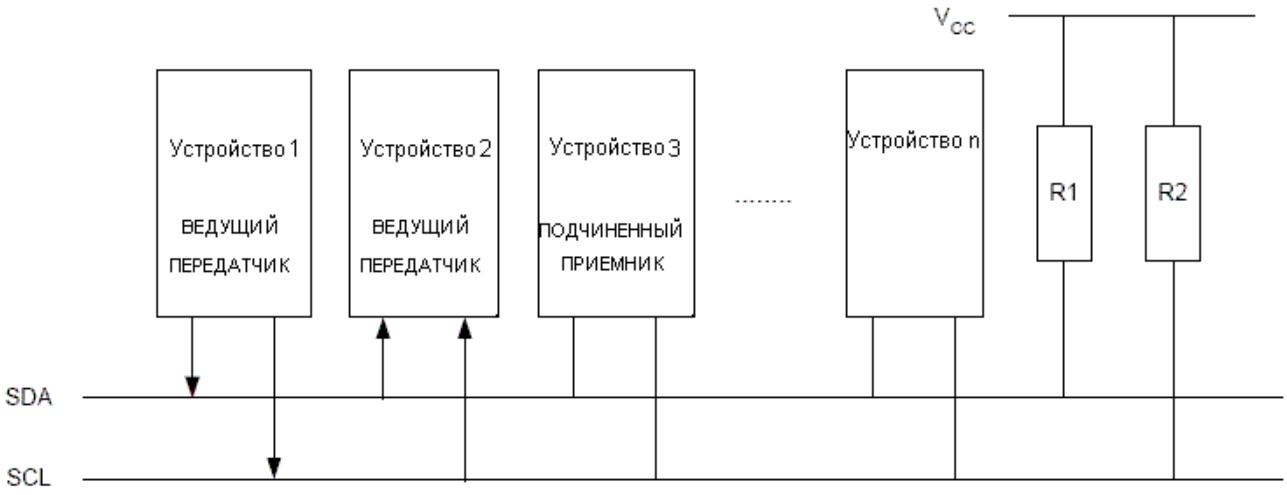


Рисунок 105. Пример арбитрации

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 106. Возможные коды состояний представлены внутри окружностей.

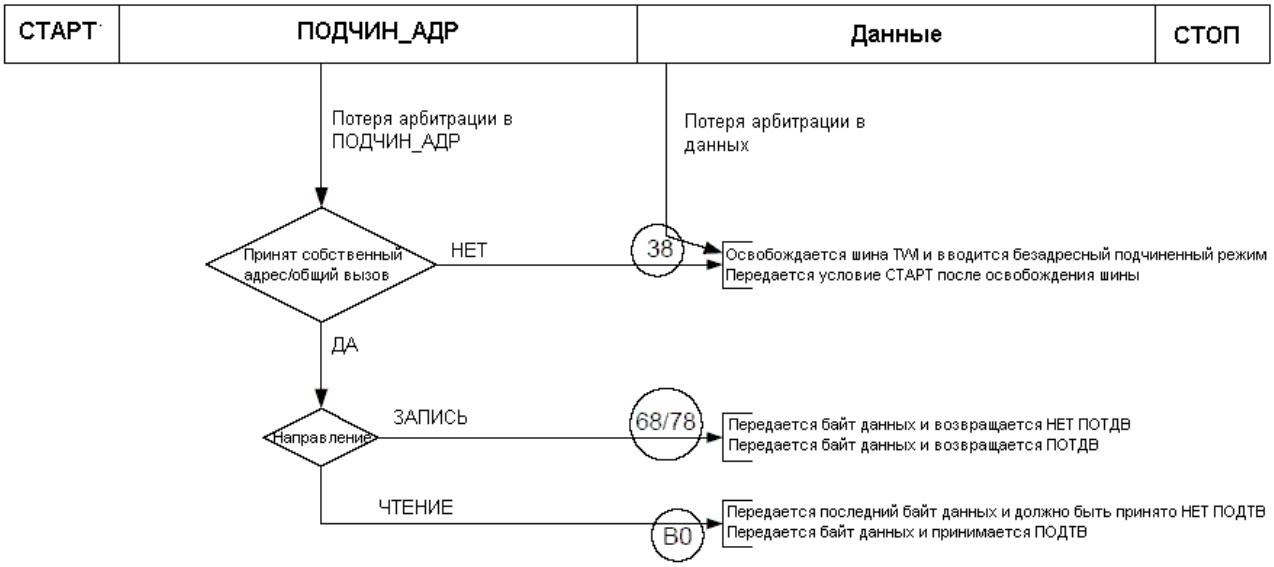


Рисунок 106. Возможные коды состояний при потере арбитрации

Программирование памяти

Биты защиты памяти программ и данных

ATmega128 содержит 6 битов защиты, которые можно оставить в незапрограммированном состоянии ("1") или же запрограммировать ("0") для активизации дополнительных функций, представленных в таблице 117. Стирание бит защиты (установка "1") может быть выполнена только командой стирание кристалла (Chip Erase).

Таблица 116. 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

ATmega128 имеет три конфигурационных байта. Таблицы 118 - 120 кратко описывают функционирование и расположение всех конфигурационных бит. Обратите внимание, что если конфигурационный бит запрограммирован, то при его считывании возвращается лог. 0.

Таблица 118. Расширенный конфигурационный байт

Наименование бита	Разряд	Описание	Исходное значение
-	7	-	1
-	6	-	1
-	5	-	1
-	4	-	1
-	3	-	1
-	2	-	1
M103C[1]	1	Режим совместимости с ATmega103	0 (запрограммированное)
WDTON[2]	0	Активизация сторожевого таймера	1 (незапрограммированное)

Прим.:

1. См. "Совместимость ATmega103 и ATmega128".
2. См. "Регистр управления сторожевым таймером - WDTCR".

Таблица 119. Старший конфигурационный байт

Наименование бита	Разряд	Описание	Исходное значение
OCDEN[4]	7	Включение встроенного блока отладки	1 (незапрограммированное, функция встроенной отладки отключена)
JTAGEN[5]	6	Включение JTAG-интерфейса	0 (запрограммированное, JTAG включен)
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 ЭСППЗУ командой стирание кристалла	1 (незапрограммированное, стирание кристалла вызывает стирание ЭСППЗУ)
BOOTSZ1	2	Выбор размера загрузочного сектора (см. табл. 113)	0 (запрограммированное) [3]
BOOTSZ0	1	Выбор размера загрузочного сектора (см. табл. 113)	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 113.
4. Не забудьте отключить бит OCDEN перед поставкой готового изделия заказчику, независимо от того какие установки имеют биты защиты и конфигурационный бит JTAGEN. Если бит OCDEN будет запрограммирован, то некоторые части системы синхронизации микроконтроллера останутся в работе при переводе микроконтроллера в экономичные режимы командой sleep. В этом случае микроконтроллер будет потреблять повышенную мощность.
5. Если интерфейс JTAG оставлен неподключенным, то конфигурационный бит JTAGEN должен быть по возможности отключен. Это позволит избежать статический ток через вывод TDO JTAG-интерфейса.

Таблица 120. 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 14.
2. Исходные установки CKSEL3..0 соответствуют выбору внутреннего RC-генератора частотой 1 МГц. Подробности представлены в таблице 6.

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 ATmega128 сигнатурными байтами являются:

1. $000: $1E (идентифицирует производителя: Atmel)
2. $001: $97 (идентифицирует размер флэш-памяти: 128 кбайт)
3. $002: $02 (идентифицирует тип микроконтроллера: ATmega128, если байт $001 равен $97)

Калибровочный байт

Внутри ATmega128 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").

Параметры параллельного программирования, расположение выводов и команды

В данном разделе описывается как у ATmega128 запрограммировать и проверить флэш-память, ЭСППЗУ, биты защиты и конфигурационные биты. Полагается, что длительность импульсов не менее 250 нс, если не имеется других указаний.

Наименование сигналов

На рисунке 135 и в таблице 121 показывается расположение и назначение выводов ATmega128, которые используются для параллельного программирования. Выводы XA1/XA0 определяют выполняемое действие при положительном фронте на выводе XTAL1 (см. табл. 123).

Подачей импульсов на WR или OE в зависимости от загруженной команды определяется выполняемое действие. Описание команд представлено в таблице 124.

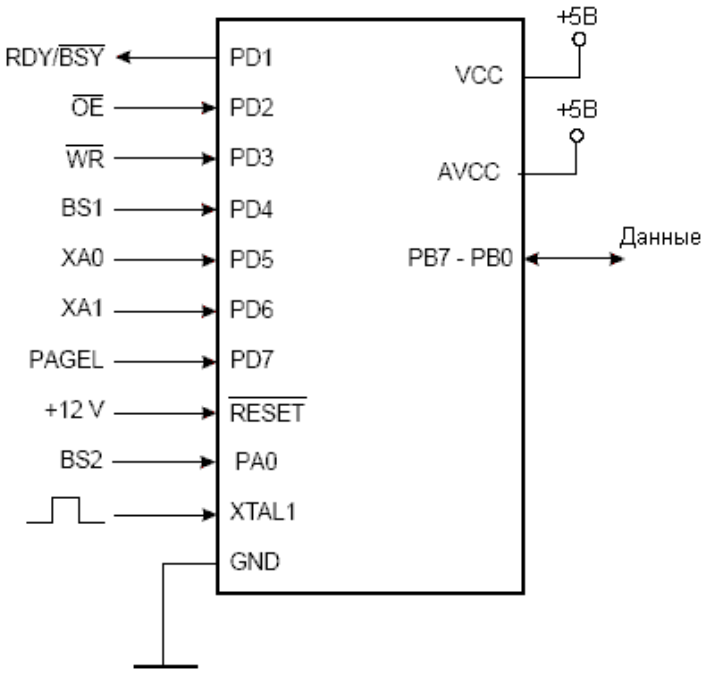


Рисунок 135. Параллельное программирование

Таблица 121. 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	PA0	Вход	Выбор байта 2 ("0" выбирает младший байт, "1" выбирает 2-ой старший байт)
DATA	PB7-0	Вход/выход	Двунаправленная шина данных (выводит данные, когда OE=0)

Таблица 122. 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 123. 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 124. 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

Таблица 125. Количество слов в странице и количество страниц во флэш-памяти

Размер флэш-памяти	Размер страницы	PCWORD	Кол. страниц	PCPAGE	PCMSB
64 кслов (128 кбайт)	128 слов	PC[6:0]	512	PC[15:7]	15

Таблица 126. Количество слов в странице и количество страниц в ЭСППЗУ

Размер ЭСППЗУ	Размер страницы	PCWORD	Кол. страниц	PCPAGE	EEAMSB
4 кбайт	8 байт	EEA[2:0]	512	EEA[11:3]	8

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. 122) 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 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 ЭСППЗУ (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 ЭСППЗУ. Данное распространяется также на чтение сигнатурных байт.

Стирание кристалла

Выполнение команды стирания кристалла приводит к очистке содержимого флэш-памяти и ЭСППЗУ (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 ЭСППЗУ.

Прим.:

1. Память ЭСППЗУ 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. 124). 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 137).
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 136. 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.
G. 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).
H. Повторяем B-H до завершения программирования всей флэш-памяти.
I. 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.

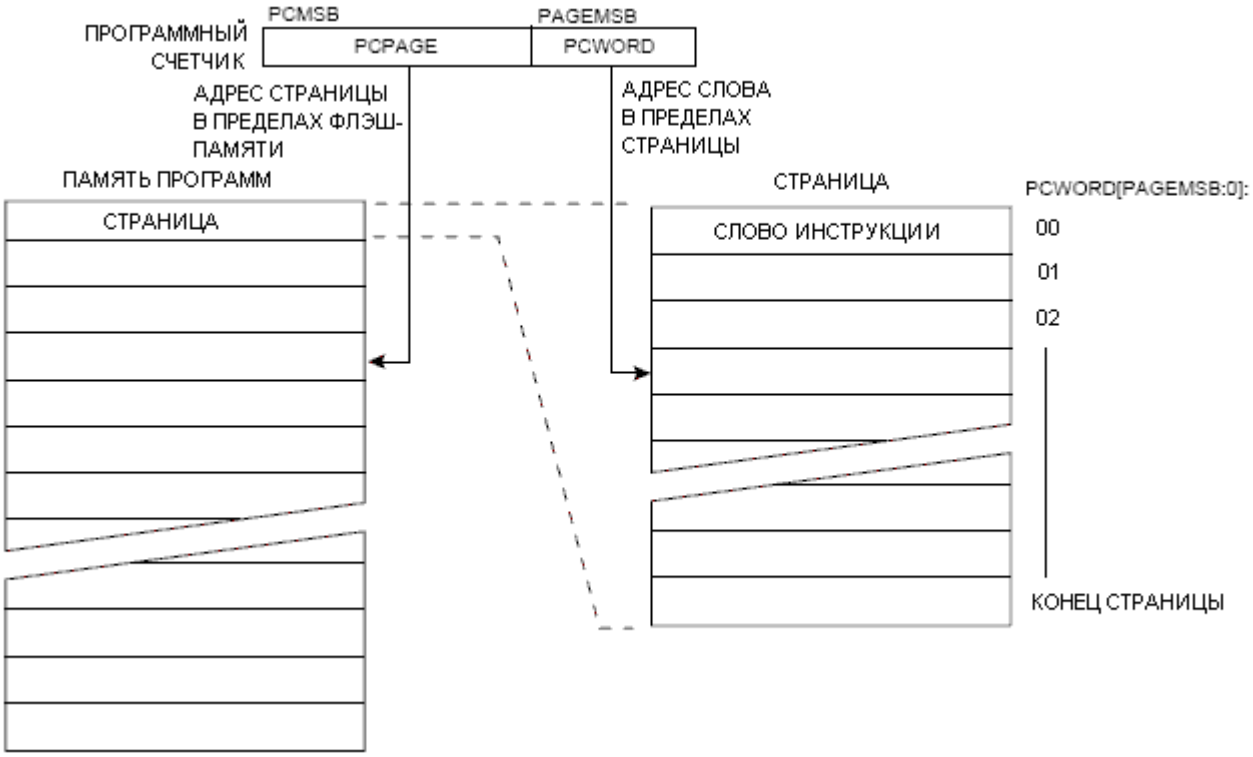


Рисунок 136. Адресация флэш-памяти со страничной организацией

Прим.:

1. PCPAGE и PCWORD представлены в таблице 125.

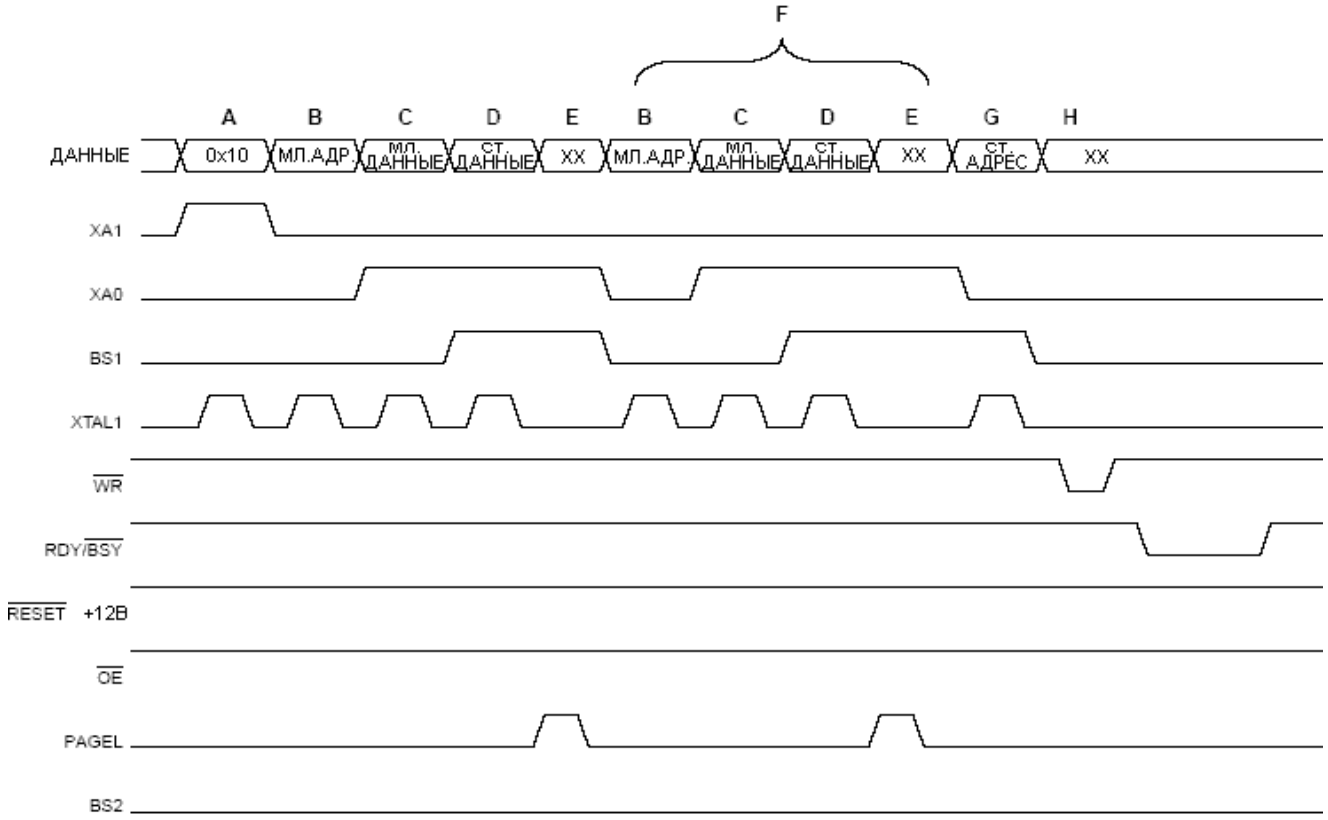


Рисунок 137. Осциллограммы сигналов программирования флэш-памяти

Прим.: "XX" означает, что не имеет значения, какие данные будут присутствовать. Указанные на рисунке символы соответствуют рассмотренному выше алгоритму.

Программирование ЭСППЗУ

ЭСППЗУ имеет страничную организацию (см. табл. 125). Во время программирования ЭСППЗУ программируемые данные размещаются в страничном буфере. Такая организация позволяет записать сразу одну страницу. Алгоритм программирования памяти данных ЭСППЗУ 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 ЭСППЗУ:
   1. Установка BS1 = "0".
   2. Формируем отрицательный импульс на WR. Этим инициируется программирование страницы ЭСППЗУ. RDY/BSY переходит в низкое состояние.
   3. Ожидаем появление лог. 1 на RDY/BSY перед программированием следующей страницы (см. осциллограммы на рисунок 138).

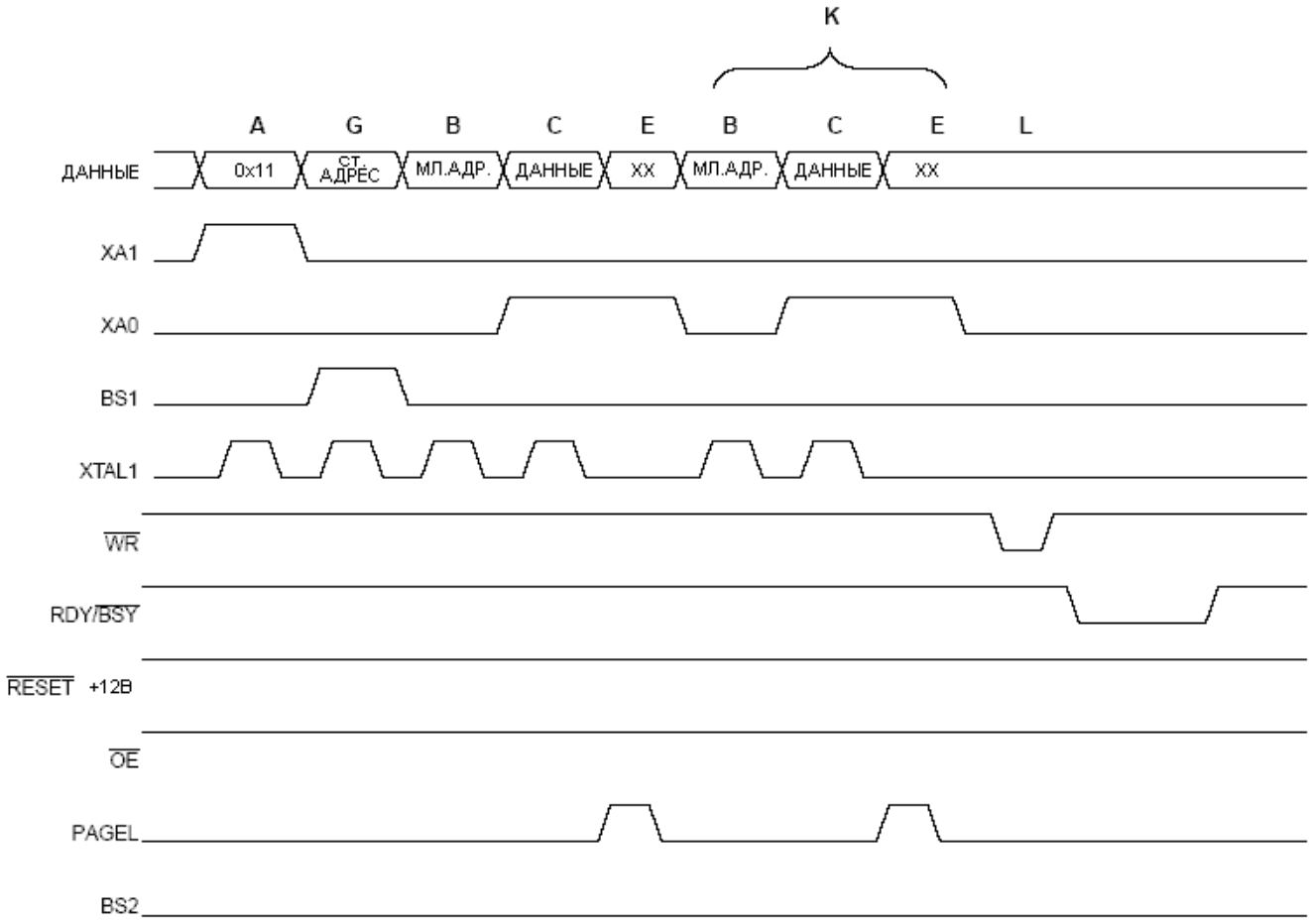


Рисунок 138. Осциллограммы сигналов программирования ЭСППЗУ

Чтение флэш-памяти

Алгоритм чтения флэш-памяти следующий (подробности по загрузке команд и адреса "Программирование флэш-памяти"):

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". С линий данных может быть считан младший байт.
5. Установка BS1 = "1". С линий данных может быть считан старший байт.
6. Установка OE = "1".

Чтение ЭСППЗУ

Алгоритм чтения ЭСППЗУ следующий (см. "Программирование флэш-памяти " для изучения подробностей загрузки команды и адреса):

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 ЭСППЗУ 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 расширенных конфигурационных бит

Алгоритм программирования расширенных конфигурационных бит описано ниже(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 BS2 = "1" и BS1 = "0". Этим выбирается расширенный байт данных.
4. Формируем отрицательный фронт на входе WR и ожидаем появление лог. 1 на RDY/BSY.
5. Устанавливаем BS2 = "0". Этим выбирается младший байт данных.

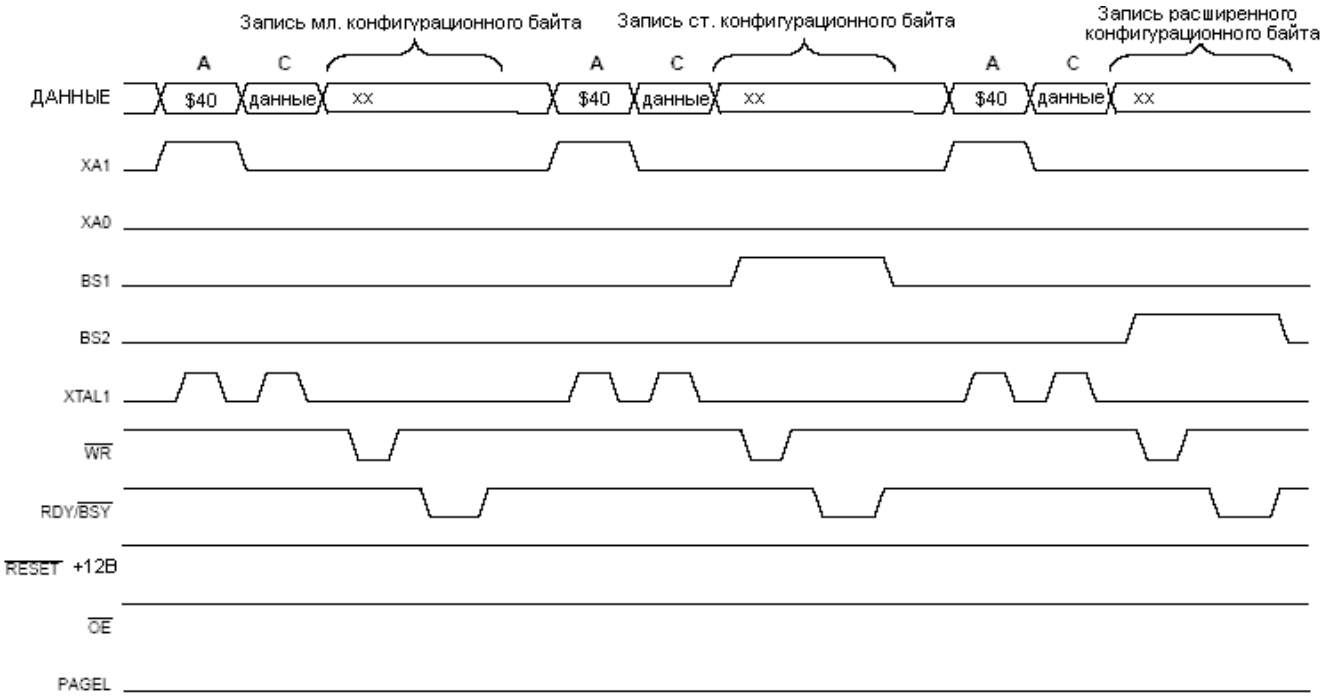


Рисунок 139. Программирование конфигурационных бит

Программирование бит защит

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 = "1" и BS1 = "0". Состояние расширенных конфигурационных бит может быть считано с линий данных ("0" означает запрограммированное состояние).
5. Установка OE = "0", BS2 = "0" и BS1 = "1". Состояние бит защиты может быть считано с линий данных ("0" означает запрограммированное состояние).
6. Установка OE = "1".

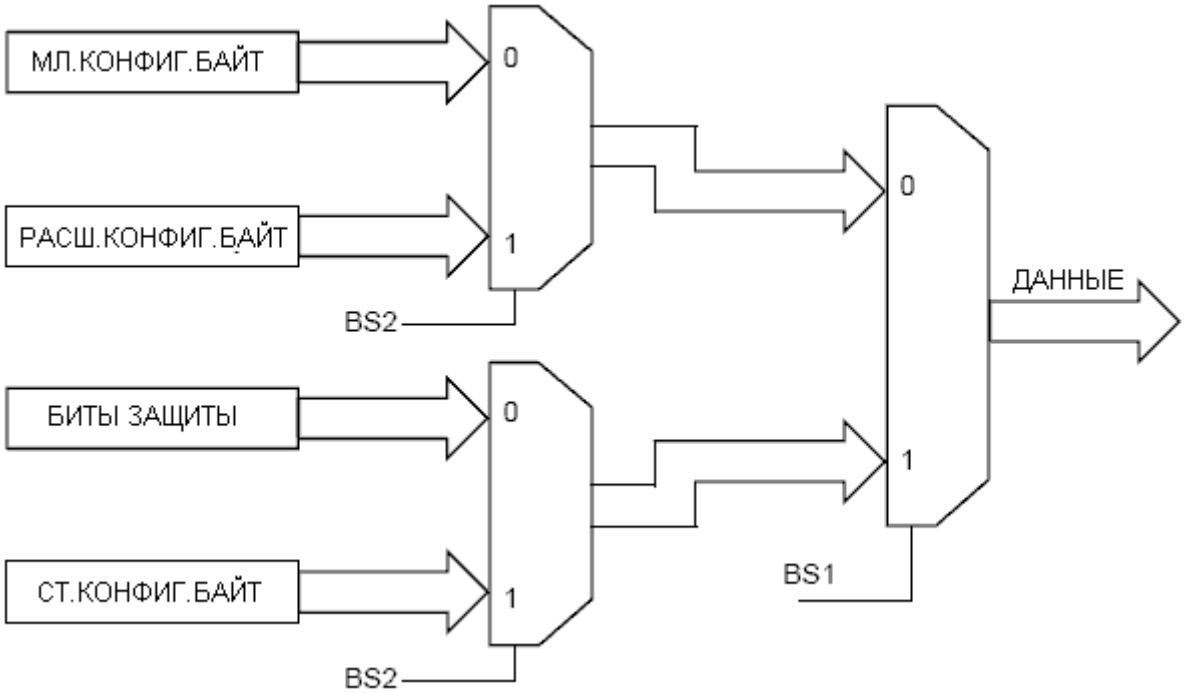


Рисунок 140. Схема считывания конфигурационных бит и бит защиты под управлением сигналов BS1, BS2

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

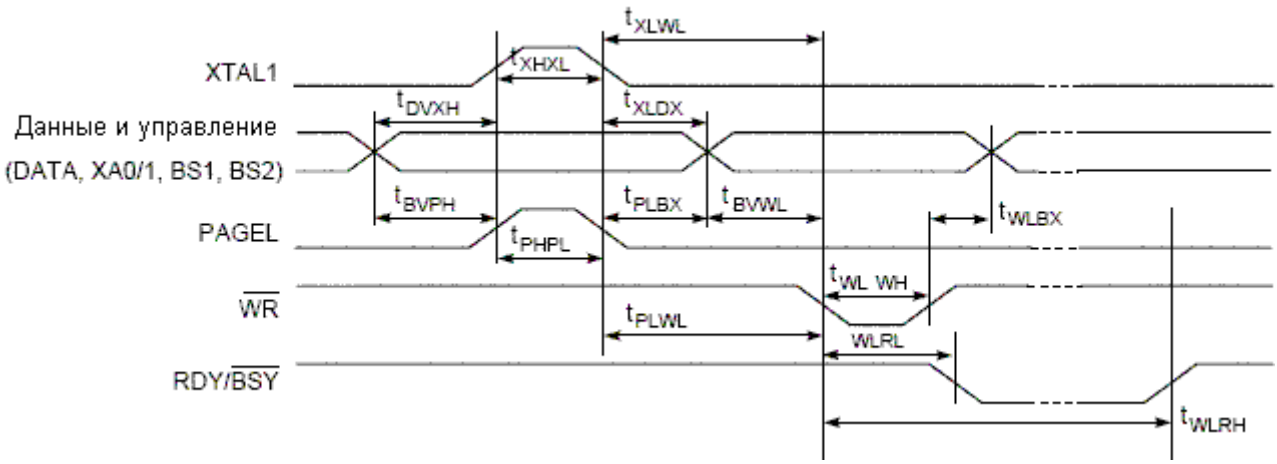


Рисунок 141. Временная диаграмма параллельного программирования: общие требования к временной диаграмме

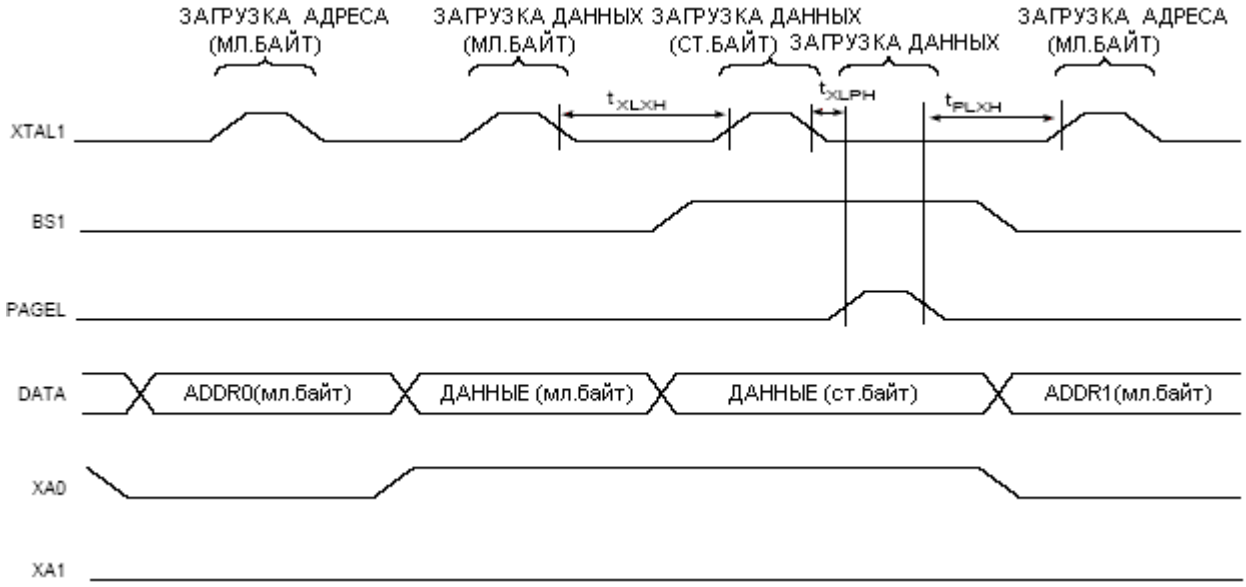


Рисунок 142. Временная диаграмма параллельного программирования: последовательность загрузки

Прим.: требования к временной диаграмме, показанные на рисунке 141 (в т.ч. tDVXH, tXHXL и tXLDX), также применимы и к операции загрузки.

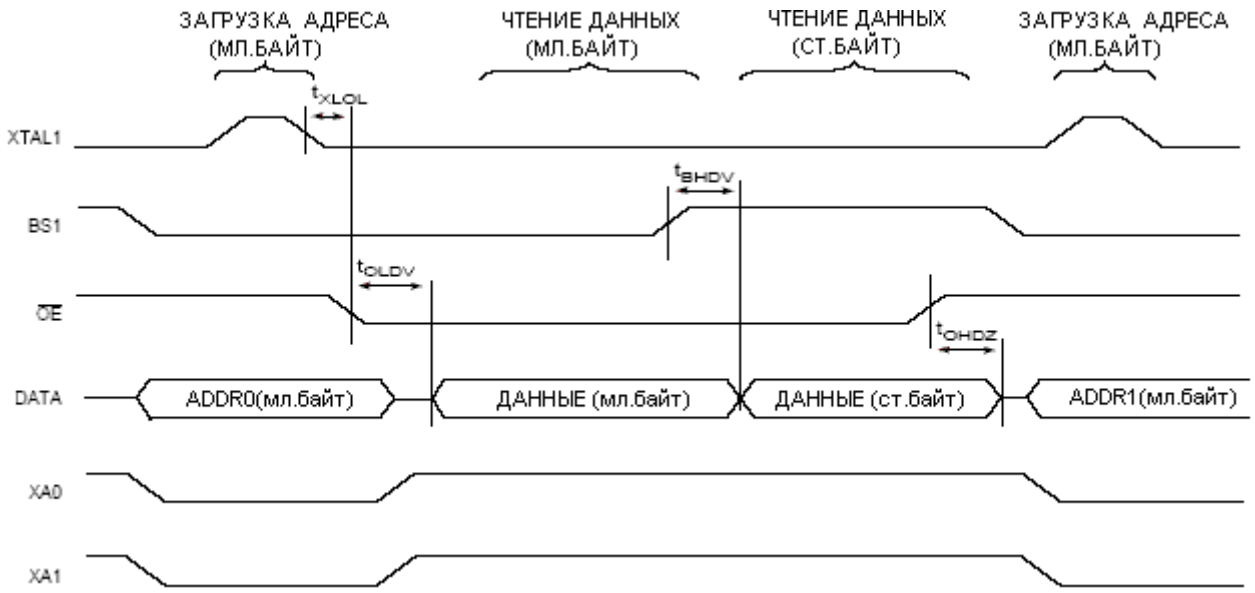


Рисунок 143. Временная диаграмма параллельного программирования: последовательность чтения (в пределах одной страницы)

Прим.: требования к временной диаграмме, показанные на рисунке 141 (в т.ч. tDVXH, tXHXL и tXLDX), также применимы и к операции загрузки.

Таблица 127. 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. t_{WLRH} относится к командам записи флэш-памяти, ЭСППЗУ, конфигурационных бит и бит защиты.
2. t_{WLRH_CE} относится к команде стирания кристалла (Chip Erase).

Последовательное программирование

Флэш-память и ЭСППЗУ 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 128 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. Для ATmega128 соответствующие выводы программирования именуются PDI и PDO.

Расположение выводов последовательного программирования через SPI

Несмотря на то, что при последовательном программирования используется тот же модуль SPI, что и при обычной работы микроконтроллера, имеется одно важное отличие: выводы MOSI/MISO модуля ввода-вывода SPI, которые совмещены с PB2 и PB3, не используются при программировании. Вместо них используются PE0 и PE1 для ввода и вывода данных при последовательном программировании (см. табл. 128).

Таблица 128. Выводы интерфейса SPI при последовательном программировании

Обозначение	Вывод	Направление	Описание
MOSI (PDI)	PE0	ввод	Последовательный ввод данных
MISO (PDO)	PE1	вывод	Последовательный вывод данных
SCK	PB1	ввод	Синхронизация последовательной связи

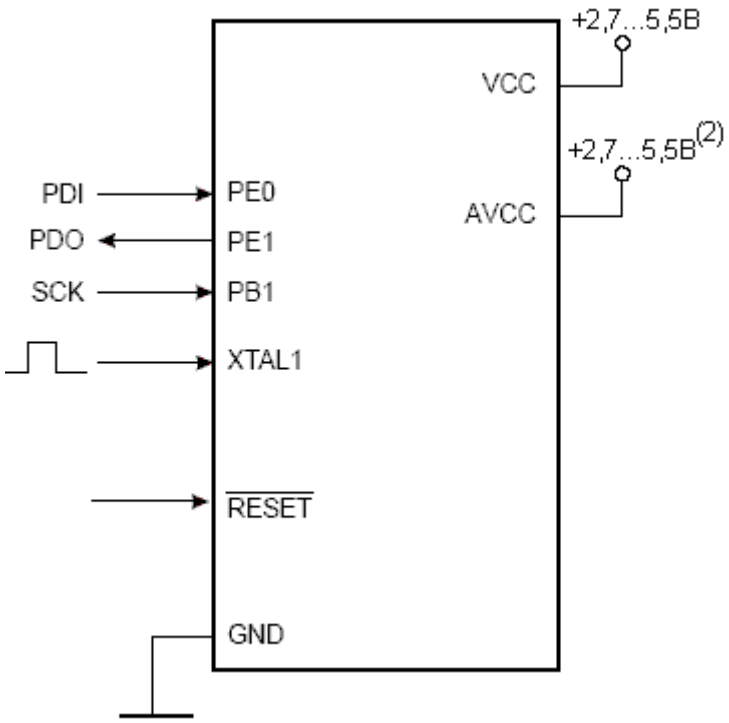


Рисунок 144. Последовательное программирование и проверка(1)

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 ЭСППЗУ 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 ЭСППЗУ 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 ≥ 12 МГц.

Длительность высокого уровня:
- больше двух тактов ЦПУ, если fck < 12 МГц;
- больше трех тактов ЦПУ, если fck ≥ 12 МГц.

Алгоритм последовательного программирования через SPI

Во время последовательной записи в ATmega128 данные тактируется нарастающим фронтом SCK. Во время чтения данных из ATmega128 данные тактируются падающим фронтом SCK. Временная диаграмма представлена на рисунке 145.

Для программирования и проверки памяти ATmega128 в режиме последовательного программирования через SPI рекомендуется придерживаться следующей последовательности (см. четырехбайтный формат в таблице 145):

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. Альтернативно сигналу RESET можно использовать вывод PEN. В этом случае будет важно только значение PEN во время сброса при подаче питания. Если программатор не гарантирует, что при подаче питания SCK =0, то использование PEN недопустимо. При использовании данного метода микроконтроллер может вернутся к нормальному режимы работы только снятием и возобновлением питания.

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 125. 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. 129).

   Прим.: если какая-либо другая команда, кроме опроса (чтения), вводится перед завершением любой операции записи (флэш-память, ЭСППЗУ, биты защиты, конфигурационные биты) программирование может завершиться некорректно.

5. Массив памяти ЭСППЗУ 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. 129). 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 129.

Опрос данных ЭСППЗУ

При чтении значения по адресу, который использовался для записи нового байта и последующего его программирования в ЭСППЗУ, 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 ЭСППЗУ 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 129 представлен о значение tWD_EEPROM.

Таблица 129. Минимальные длительности задержек перед записью очередной ячейки флэш-памяти и ЭСППЗУ

Обозначение	Минимальная задержка
tWD_FLASH	4.5 мс
tWD_EEPROM	9.0 мс
tWD_ERASE	9.0 мс

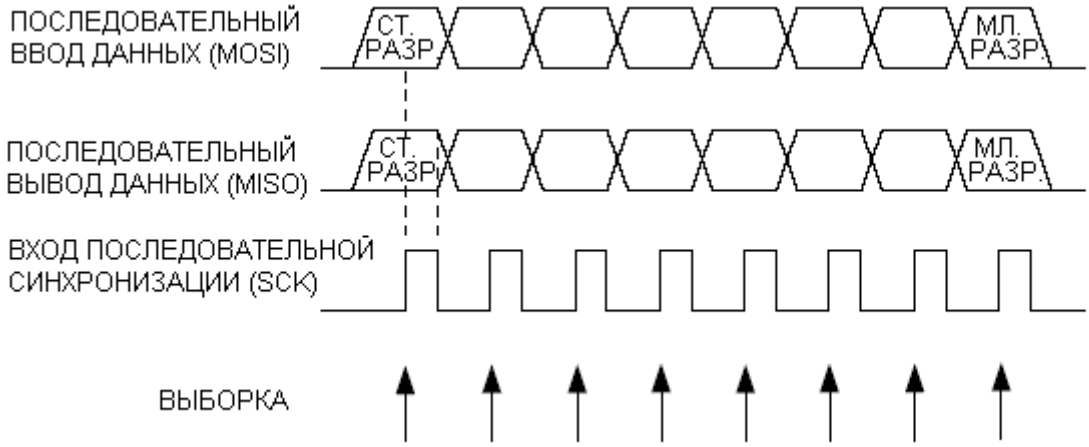


Рисунок 145. Осциллограммы сигналов последовательного программирования интерфейса SPI

Таблица 130. 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 ЭСППЗУ и флэш-памяти
Чтение памяти программ	0010 H000	aaaa 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 H000	xxxx xxxx	xbbb 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	aaaa aaaa	bxxx xxxx	xxxx xxxx	Запись страницы памяти программ по адресу a:b.
Чтение ЭСППЗУ	1010 0000	xxxx aaaa	bbbb bbbb	oooo oooo	Чтение данных o из ЭСППЗУ по адресу a:b.
Запись ЭСППЗУ	1100	xxxx	bbbb	iiii iiii	Запись данных i в ЭСППЗУ по адресу a:b.

	0000	aaaa	bbbb		
Чтение бит защиты	0101 1000	0000 0000	xxxx xxxx	xxoo oooo	Чтение бит защиты. "0" - запрограммирован, "1" - не запрограммирован. См. табл. 116.
Запись бит защиты	1010 1100	111x xxxx	xxxx xxxx	11ii iiii	Запись бит защиты. Запись "0" приводит к программированию бита защиты. См. табл. 116.
Чтение сигнатурного байта	0011 0000	xxxx xxxx	xxxx xxbb	oooo oooo	Чтение сигнатурного байта o по адресу b.
Запись конфигурационных бит	1010 1100	1010 0000	xxxx xxxx	iiii iiii	Указывайте "0" для программирования, "1" для стирания. См. табл. 120.
Запись старших конфигурационных бит	1010 1100	1010 1000	xxxx xxxx	iiii iiii	Указывайте "0" для программирования, "1" для стирания. См. табл. 120.
Запись расширенных конфигурационных бит	1010 1100	1010 0100	xxxx xxxx	xxxx xxii	Указывайте "0" для программирования, "1" для стирания. См. табл. 120.
Чтение конфигурационных бит	0101 0000	0000 0000	xxxx xxxx	oooo oooo	Чтение конфигурационных бит. "0" - запрограммирован, "1" - не запрограммирован. См. табл. 120.
Чтение расширенных конфигурационных бит	0101 0000	0000 1000	xxxx xxxx	oooo oooo	Чтение расширенных конфигурационных бит. "0" - запрограммирован, "1" - не запрограммирован. См. табл. 120.
Чтение старших конфигурационных бит	0101 1000	0000 1000	xxxx xxxx	oooo oooo	Чтение старших конфигурационных бит. "0" = запрограммирован, "1" = не запрограммирован. См. табл. 119.
Чтение калибровочного байта	0011 1000	xxxx xxxx	0000 00bb	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**".

Программирование через интерфейс JTAG

Для программирования через интерфейс JTAG требуется использовать четыре специфических вывода JTAG-интерфейса: TCK, TMS, TDI и TDO. Управление выводами сброса и тактирования микроконтроллера не требуется. Для активизации JTAG-интерфейса необходимо запрограммировать конфигурационный бит JTAGEN. В состоянии поставки у микроконтроллера данный бит запрограммирован. Кроме этого, необходимо сбросить бит JTD в регистре MCUCSR. Альтернативно бит JTD можно сбросить путем удержания входа сброса в низком состоянии в течении двух тактов ЦПУ, после чего выводы JTAG-интерфейса доступны для программирования.

Этим обеспечивается возможность использовать выводы JTAG-интерфейса в качестве линий ввода-вывода в процессе нормальной работы, а при необходимости выполнить внутрисистемное программирование через JTAG-интерфейс. Обратите внимание, что данный метод не может использоваться для граничного сканирования или внутренней отладки через выводы JTAG-интерфейса. В таком случае выводы JTAG-интерфейса должны использоваться только для этих целей. Также необходимо выделить, что, как при последовательной передачи, так и при последовательном приеме, первым передается младший разряд всех сдвиговых регистров.

Программирование специфических JTAG-инструкций

Регистр инструкции является 4-х разрядным и, поэтому, поддерживает 16 инструкций. JTAG-инструкции, полезные для программирования, представлены ниже.

Код операции каждой инструкции показан за наименованием инструкции в 16-тиричном формате. Текстом описывается, какой регистр данных выбирается в качестве пути между TDI и TDO для каждой инструкции.

Состояние "тест-старт/свободен" TAP-контроллера используется для генерации внутренней синхронизации. Он может также использоваться как промежуточное состояние "простоя" между JTAG-последовательностями. Граф-автомат для изменения слова инструкции на рисунке 146.

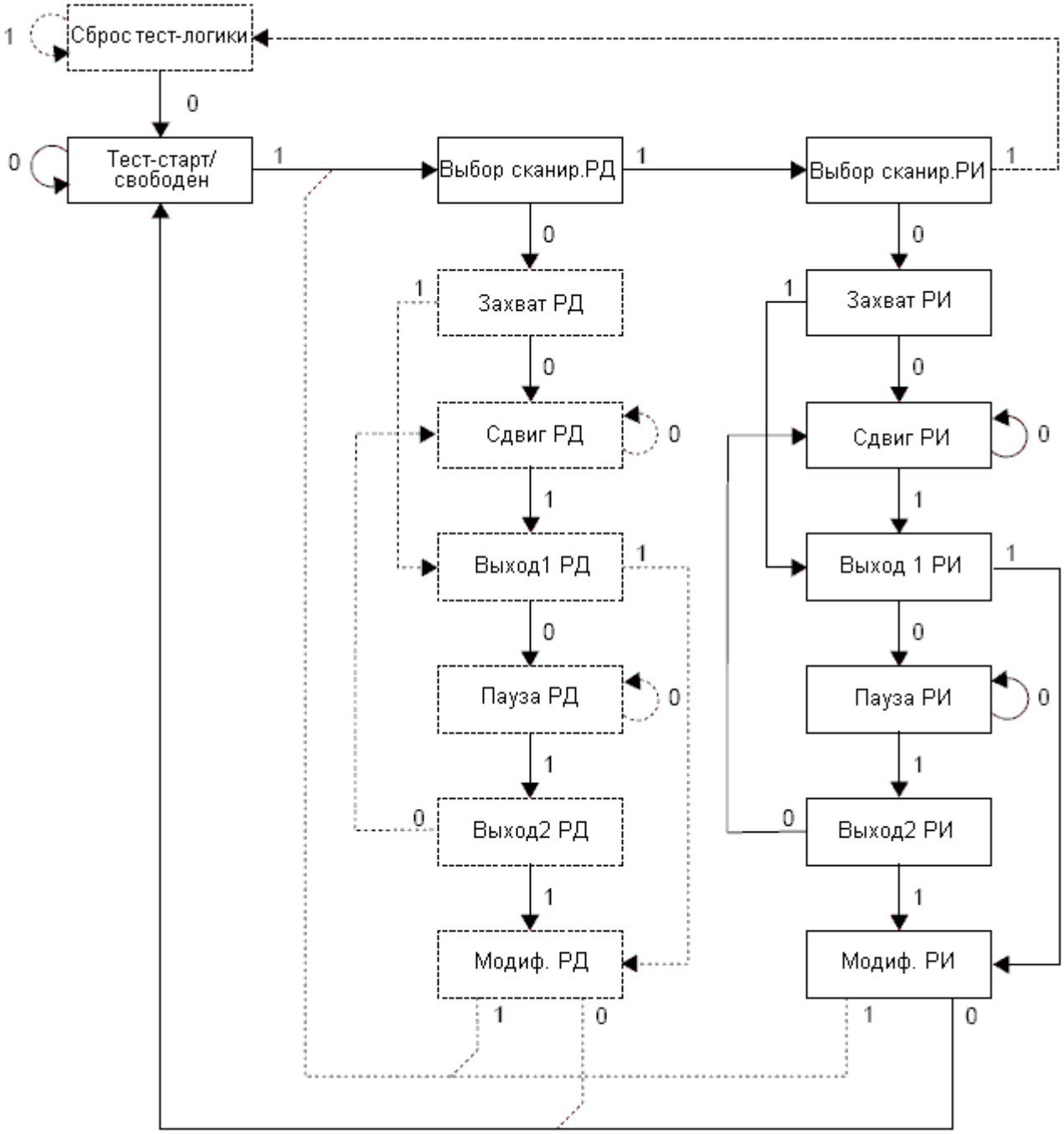


Рисунок 146. Граф-автомат изменения слова инструкции

AVR_RESET; $C

Специфическая инструкция, которая поддерживается AVR-микроконтроллерами, для принудительного перевода микроконтроллера в состояние сброса. TAP-контроллер не

сбрасывается при выполнении данной инструкции. В качестве регистра данных выступает одноразрядный регистр сброса. Обратите внимание, что микроконтроллер будет находиться непрерывно в состоянии сброса, пока в регистре сброса (и цепи сканирования) будет записана лог. 1. Выход данной цепи не защелкивается.

Активными состояниями являются:

СДВИГ РД: Регистр сброса сдвигается под управлением входа TCK.

PROG_ENABLE ($4)

Специфическая инструкция для разрешения программирования через JTAG-порт. 16-разр. регистр разрешения программирования выбирается в качестве регистра данных. Активными состояниями являются следующие:

СДВИГ РД: сигнатурный код разрешения программирования загружается в регистр данных.
МОДИФ РД: сравнение с сигнатурным кодом разрешения программирования и ввод режима программирования при обнаружении совпадения.

PROG_COMMANDS ($5)

Специфическая инструкция для ввода команд программирования через порт JTAG. 15-разр. регистр команд программирования выбирается в качестве регистра данных. Активными состояниями являются:

ЗАХВАТ РД: результат предыдущей команды загружается в регистр данных.
СДВИГ РД: регистр данных сдвигается под управлением тактового входа TCK, сдвигом выводится результат предыдущей команды и вводится новая команда.
МОДИФ_РД: команды программирования поступает на входы флэш-памяти.
ТЕСТ-СТАРТ/СВОБОДЕН: генерируется один такт синхронизации, выполняется загруженная команда (не всегда требуется, см. табл. 131).

PROG_PAGELOAD ($6)

Специфическая JTAG-инструкция, которая непосредственно загружает страницу данных флэш-памяти через JTAG-порт. 2048-битный виртуальный регистр загрузки страницы флэш-памяти выбирается в качестве регистра данных. Он является виртуальной цепью сканирования с длиной равной количеству бит в странице флэш-памяти. Внутренний сдвиговый регистр - 8-разрядный. В отличие от большинства JTAG-инструкций состояние МОДИФ РД не используется для передачи данных из сдвигового регистра. Данные автоматически передаются в страничный буфер флэш-памяти побайтно в состоянии СДВИГ РД.

Активными состояниями являются:

СДВИГ РД: страница данных флэш-памяти вводится через TDI под управлением TCK и автоматически загружается в страницу флэш-памяти побайтно.

PROG_PAGEREAD ($7)

Специфическая JTAG-инструкция полного считывания одной страницы флэш-памяти через JTAG-порт. 2056-разрядный виртуальный регистр чтения страницы флэш-памяти выбирается в качестве регистра данных. Он является виртуальной цепью сканирования с длиной эквивалентной количеству бит одной страницы флэш-памяти + 8 бит. Внутренний сдвиговый регистр является 8-ми разрядным. В отличие от большинства JTAG-инструкций состояние МОДИФ РД не используется для передачи данных из сдвигового регистра. Данные автоматически передаются из страничного буфера флэш-памяти побайтно в состоянии СДВИГ РД.

Активными состояниями являются:

СДВИГ РД: автоматически считывается один байт данных из флэш-памяти и передается сдвигом через TDO под управлением входа TCK. Вход TDI игнорируется.

Регистры данных

Регистры данных выбираются регистрами JTAG-инструкций, как описано "Специфические JTAG-инструкции программирования". Ниже перечислены регистры данных, которые относятся к операциям программирования:

Регистр сброса
Регистр разрешения программирования
Регистр команд программирования
Виртуальный регистр загрузки флэш-памяти
Виртуальный регистр чтения флэш-памяти

Регистр сброса

Регистр сброса является тестовым регистром данных и используется для сброса микроконтроллера в процессе программирования. Такой сброс требуется перед вводом режима программирования. Запись лог. 1 в регистр сброса приводит к установке низкого уровня на внешнем входе сброса "Reset". Микроконтроллер будет находиться в состоянии сброса до тех пор, пока регистр сброса имеет единичное состояние. В зависимости от настроек синхронизации конфигурационными битами после записи лог.0 в регистр сброса микроконтроллер задержится в состоянии сброса в течение фиксированного промежутка времени (см. также "Источники синхронизации"). Выход из данного регистра данных не защелкивается, т.к. сброс наступает незамедлительно, как показано на рисунке 123.

Регистр разрешения программирования

Регистр разрешения программирования является 16-разрядным. Содержимое этого регистра сравнивается с сигнатурным кодом разрешения программирования "1010_0011_0111_0000". Если содержимое регистра совпадает с сигнатурным кодом разрешения программирования, то разрешается программирование через порт JTAG. Регистр принимает нулевое состояние после сброса при подаче питания и должен быть сброшен при выходе из режима программирования.

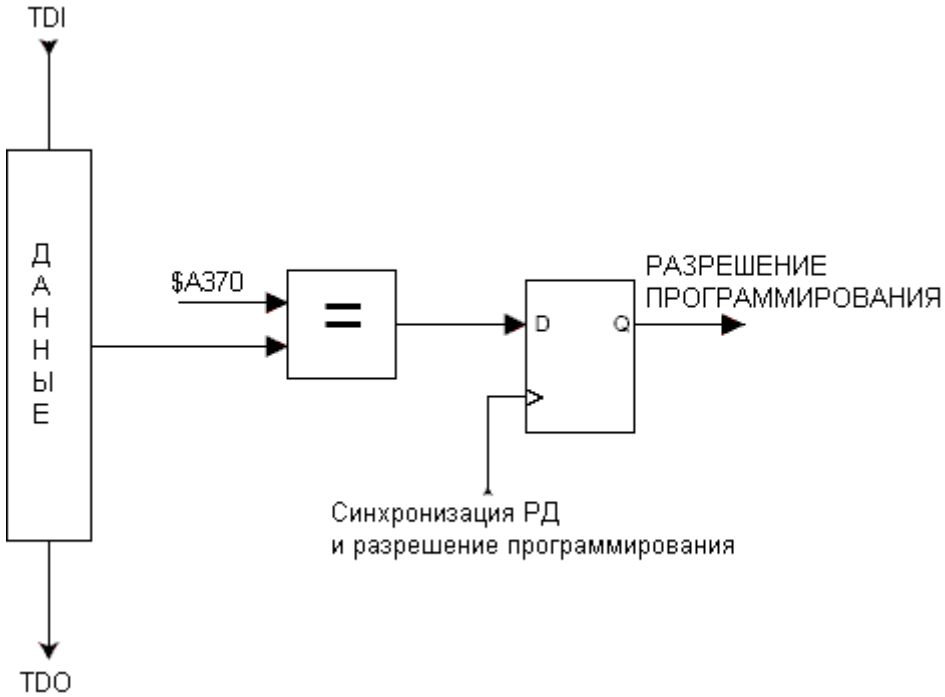


Рисунок 147. Регистр разрешения программирования

Регистр команд программирования

Регистр команд программирования является 15-разрядным регистром. Данный регистр используется для последовательной загрузки команд программирования, при этом передается результат предыдущей команды. Набор JTAG-инструкций программирования показан в таблице 131. Граф-автомат загрузки команд программирования показан на рисунке 149.

Рисунок 148. Регистр команд программирования

Таблица 131. JTAG-инструкции программирования

Инструкция	Последовательность TDI	Последовательность TDO	Примечания
1a. Стирание кристалла	0100011_10000000 0110001_10000000 0110011_10000000 0110011_10000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
1b. Опрос завершения стирания кристалла	0110011_10000000	xxxxxox_xxxxxxxx	(2)
2a. Ввод записи флэш-памяти	0100011_00010000	xxxxxxx_xxxxxxxx	
2b. Загрузка старшего адресного байта	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
2c. Загрузка младшего адресного байта	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
2d. Загрузка младшего байта данных	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
2e. Загрузка старшего байта данных	0010111_iiiiiiii	xxxxxxx_xxxxxxxx	
2f. Запись данных	0110111_00000000 1110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)

	0110111_00000000	xxxxxxx_xxxxxxxx	
2g. Запись страницы флэш-памяти	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxx xxxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
2h. Опрос завершения записи страницы флэш-памяти	0110111_00000000	xxxxxox_xxxxxxxx	(2)
3a. Ввод чтения флэш-памяти	0100011_00000010	xxxxxxx_xxxxxxxx	
3b. Загрузка старшего адресного байта	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
3c. Загрузка младшего адресного байта	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
3d. Чтение младшего и старшего байта данных	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo xxxxxxx_oooooooo	младший байт старший байт
4a. Ввод записи ЭСППЗУ	0100011_00010001	xxxxxxx_xxxxxxxx	
4b. Загрузка старшего адресного байта	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
4c. Загрузка младшего адресного байта	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
4d. Загрузка байта данных	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
4e. запись данных	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4f. Запись страницы ЭСППЗУ	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
4g. Опрос завершения страничной записи	0110011_00000000	xxxxxox_xxxxxxxx	(2)
5a. Ввод чтения ЭСППЗУ	0100011_00000011	xxxxxxx_xxxxxxxx	
5b. Загрузка старшего адресного байта	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	(9)
5c. Загрузка младшего адресного байта	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
5d. Чтение байта данных	0110011_bbbbbbbb 0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
6a. Ввод записи конфигурационных бит	0100011_01000000	xxxxxxx_xxxxxxxx	
6b. Чтение младшего байта данных [6]	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	(3)
6c. Запись расширенного конфигурационного байта	0111011_00000000 0111001_00000000 0111011_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6d. Опрос завершения записи расширенного конфигурационного байта	0110111_00000000	xxxxxox_xxxxxxxx	(2)
6e. Загрузка младшего байта данных [7]	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	(3)
6f. Запись старшего конфигурационного байта	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6g. Опрос завершения записи	0110111_00000000	xxxxxox_xxxxxxxx	(2)

конфигурационного байта			
6h. Загрузка младшего байта данных[7]	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	(3)
6i. Запись младшего конфигурационного байта	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
6j. Опрос завершения записи конфигурационного байта	0110011_00000000	xxxxxox_xxxxxxxx	(2)
7a. Ввод записи бит защиты	0100011_00100000	xxxxxxx_xxxxxxxx	
7b. Загрузка байта данных[9]	0010011_11iiiiii	xxxxxxx_xxxxxxxx	(4)
7c. Запись бит защиты	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	(1)
7d. Опрос завершения записи бит защиты	0110011_00000000	xxxxxox_xxxxxxxx	(2)
8a. Ввод чтения бит защиты/конфигурационных бит	0100011_00000100	xxxxxxx_xxxxxxxx	
8b.Чтение расширенного конфигурационного байта [6]	0111010_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
8c. Чтение старшего конфигурационного байта [7]	0111110_00000000 0111111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
8d. Чтение младшего конфигурационного байта [8]	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
8e. Чтение бит защиты [9]	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxoooooo	(5)
8f. Чтение конфигурационных бит и бит защиты	0111010_00000000 0111110_00000000 0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo xxxxxxx_oooooooo xxxxxxx_oooooooo xxxxxxx_oooooooo	(5) расш. конф. байтстарш. конф. байтмл. конф. байтбиты защиты
9a. Ввод чтения сигнатурного байта	0100011_00001000	xxxxxxx_xxxxxxxx	
9b. Загрузка адреса байта	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
9c. Чтение сигнатурного байта	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
10a. Ввод чтения калибровочного байта	0100011_00001000	xxxxxxx_xxxxxxxx	
10b. Загрузка адреса байта	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
10c. Чтение калибровочного байта	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
11a. Загрузка команды "нет операции"	0100011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	

Сокращения:

a - адрес старших разрядов;
b - адрес младших разрядов;
H - выбор младшего (=0) или старшего (=1) байта;
o - выводимые данные;
i - вводимые данные;
x - может иметь произвольное значение.

Прим.:

1. Данная последовательность не требуется, если семь старших разрядов корректно установлены предыдущей последовательностью команд.
2. Повторить до тех пор, пока o = "1".
3. Установить биты равными "0" для программирования соответствующих конфигурационных бит и "1" для их стирания.
4. Установить биты равными "0" для программирования соответствующих бит защиты и "1", чтобы оставить их неизменными.
5. "0" -запрограммирован, "1" - не запрограммирован.
6. Информация по расположению расширенных конфигурационных бит представлена в таблице 118.
7. Информация по расположению старших конфигурационных бит представлена в таблице 119.
8. Информация по расположению младших конфигурационных бит представлена в таблице 120.
9. Информация по расположению бит защиты представлена в таблице 116.

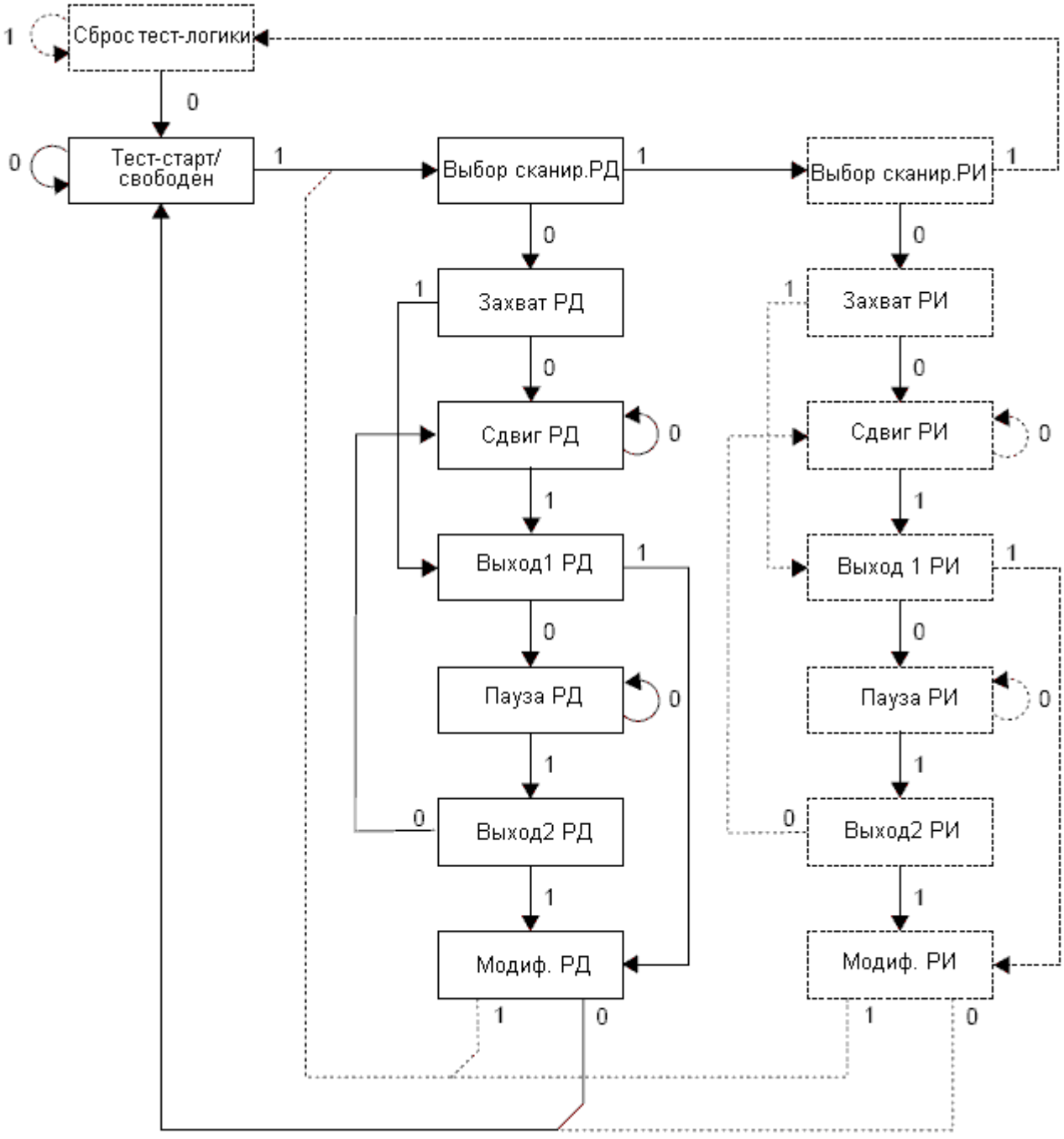


Рисунок 149. Граф-автомат изменения/чтения слов данных

Виртуальный регистр загрузки страницы флэш-памяти

Виртуальный регистр загрузки флэш-памяти представляет собой виртуальную цепь сканирования с длиной равной количеству бит в одной странице флэш-памяти. Внутренний сдвиговый регистр 8-разрядный и данные автоматически передаются в страничный буфер побайтно. Слова инструкций вводятся сдвигом, начиная с младшего разряда первого слова инструкции и завершая старшим разрядом последней инструкции в пределах одной и той же страницы. Этим обеспечивается эффективный путь загрузки страничного буфера флэш-памяти перед выполнение операции "Запись страницы".

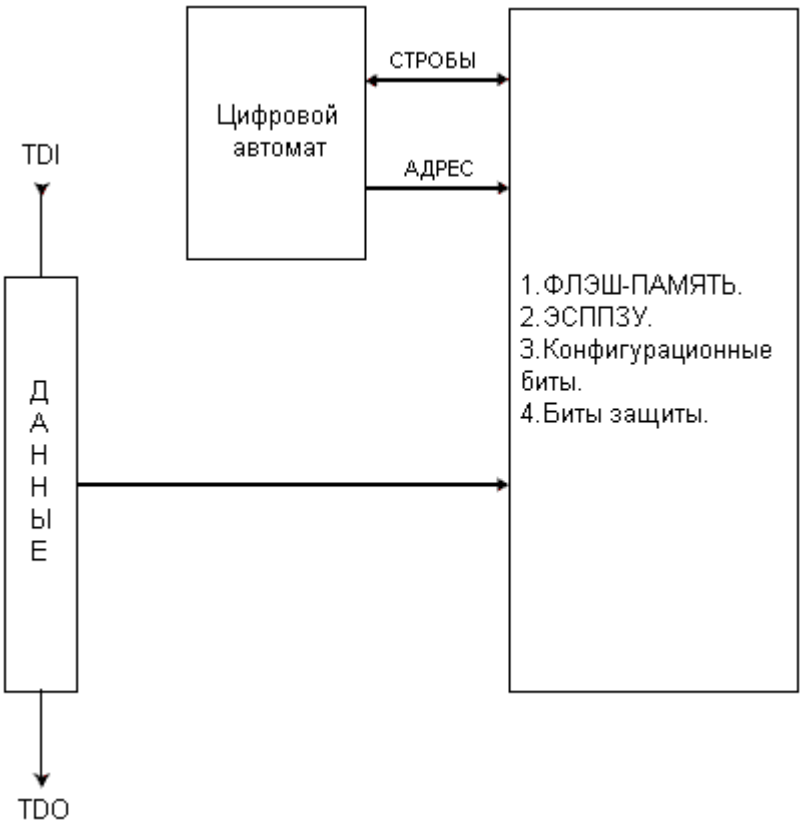


Рисунок 150. Виртуальный регистр загрузки страницы флэш-памяти

Виртуальный регистр чтения страницы флэш-памяти

Виртуальный регистр чтения страницы флэш-памяти представляет собой виртуальную цепь сканирования с длиной равной количеству бит в странице флэш-памяти + 8 бит. Фактически сдвиговый регистр 8-разрадный и данные автоматически передаются из страницы флэш-памяти побайтно. Первые 8 тактов используются для передачи первого байта в сдвиговый регистр и передаваемые в течении этих 8-ми тактов биты необходимо игнорировать. После этой инициализации данные сдвигаются, начиная с младшего разряда первой инструкции текущей страницы и заканчивая старшим разрядом последней инструкции этой же страницы. Этим обеспечивается эффективный путь чтения страницы флэш-памяти для проверки результата программирования.

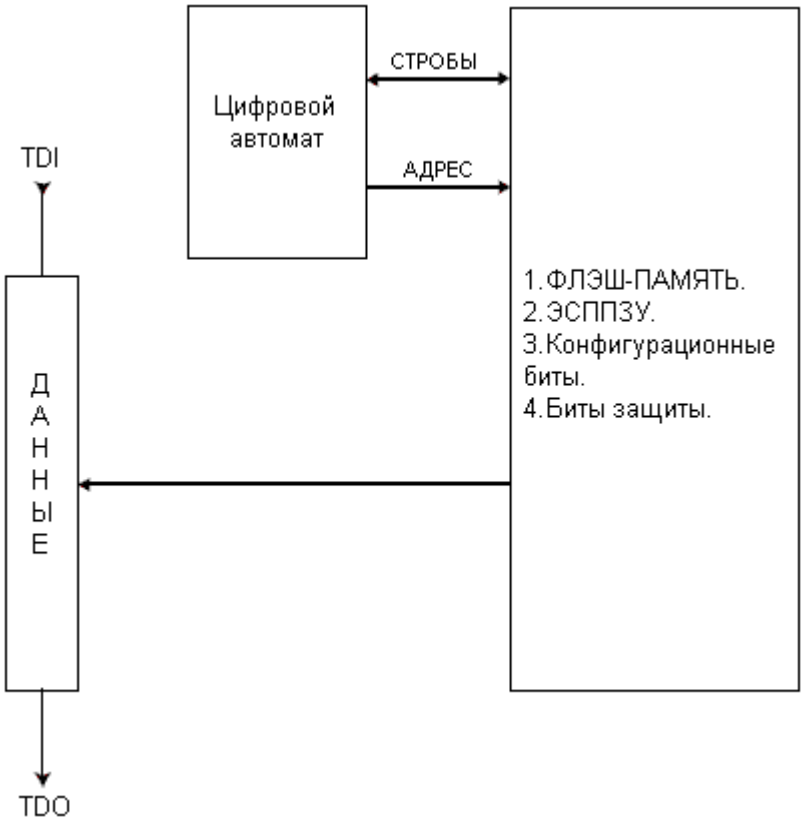


Рисунок 151. Виртуальный регистр чтения страницы флэш-памяти

Алгоритм программирования

Ниже используются ссылки на коды инструкций "1a", "1b" и т.п. Информация о них представлена в таблице 131.

Ввод режима программирования

1. Ввод JTAG-инструкции AVR_RESET и сдвиг 1 в регистр сброса.
2. Ввод инструкции PROG_ENABLE и сдвиг 1010_0011_0111_0000 в регистр разрешения программирования.

Выход из режима программирования

1. Ввод JTAG-инструкции PROG_COMMANDS.
2. Отключение всех инструкций программирования вводом инструкции "нет операции" 11a.
3. Ввод инструкции PROG_ENABLE и загрузка регистра разрешения программирования кодом 0000_0000_0000_0000.
4. Ввод JTAG-инструкции AVR_RESET и сдвиг 0 в регистр сброса.

Выполнение стирания кристалла

1. Ввод JTAG-инструкции PROG_COMMANDS.
2. Запуск стирания кристалла вводом инструкции программирования 1a.
3. Опрос завершения стирания кристалла с помощью инструкции программирования 1b или задержка на время tWLRH_CE (см. табл. 127).

Программирование флэш-памяти

Перед программированием флэш-памяти необходимо выполнить операцию "стирание кристалла" (см. "Выполнение стирания кристалла").

1. Ввод JTAG-инструкции PROG_COMMANDS.

2. Разрешение записи флэш-памяти, используя инструкцию программирования 2a.
3. Загрузка старшего байта адреса с помощью инструкции программирования 2b.
4. Загрузка младшего байта адреса с помощью инструкции программирования 2c.
5. Загрузка данных с помощью инструкций программирования 2d, 2e и 2f.
6. Повторение шагов 4 и 5 для всех слов инструкций загружаемой страницы.
7. Запись страницы с помощью инструкции программирования 2g.
8. Опрос завершения записи флэш-памяти, используя инструкцию программирования 2h или задержка на время tWLRH (см. табл. 127).
9. Повторение шагов 3…7 до завершения программирования всех данных. Более эффективная передачи данных может быть достигнута с помощью инструкции PROG_PAGELOAD:

1. Ввод JTAG-инструкции PROG_COMMANDS.
2. Разрешение записи флэш-памяти с помощью инструкции программирования 2a.
3. Загрузка адреса страницы с помощью инструкций 2b и 2c. PCWORD (см. табл. 124) используется для адресации в пределах одной страницы и в него необходимо записать 0.
4. Ввод JTAG-инструкции PROG_PAGELOAD.
5. Загрузка всей страницы путем загрузки в страницу всех слов инструкций сдвигом, начиная с младшего разряда первой инструкции текущей страницы и завершая старшим разрядом последней инструкции этой же страницы.
6. Ввод JTAG-инструкции PROG_COMMANDS.
7. Запись страницы с помощью инструкции программирования 2g.
8. Опрос завершения записи флэш-памяти с помощью инструкции программирования 2h или задержка на время tWLRH (см. табл.127).
9. Повторение шагов 3…8 до завершения программирования всех данных.

Чтение флэш-памяти

1. Ввод JTAG-инструкции PROG_COMMANDS.
2. Разрешение чтения флэш-памяти, используя инструкцию программирования 3a.
3. Загрузка адреса с помощью инструкций программирования 3b и 3c.
4. Чтение данных с помощью инструкции программирования 3d.
5. Повторение шагов 3 и 4 до завершения чтения всех данных.

Более высокая эффективность передачи данных может быть достигнута с помощью инструкции PROG_PAGEREAD:

1. Ввод JTAG-инструкции PROG_COMMANDS.
2. Разрешение чтения флэш-памяти с помощью инструкции программирования 3a.
3. Загрузка адреса страницы с помощью инструкций программирования 3b и 3c. PCWORD (см. табл. 124) используется для адресации в пределах одной страницы и должен быть записан также.
4. Ввод JTAG-инструкции PROG_PAGEREAD.
5. Чтение всей страницы путем сдвига всех слов инструкций в странице, начиная с младшего разряда первой инструкции и завершая старшим разрядом последней инструкции. Помните, что первые 8 сдвигаемых бит необходимо игнорировать.
6. Ввод JTAG-инструкции PROG_COMMANDS.
7. Повторение шагов 3…6 до завершения чтения всех данных.

Программирование ЭСППЗУ

Перед программированием ЭСППЗУ должно быть выполнено "стирание кристалла". См. "Выполнение стирания кристалла".

1. Ввод JTAG-инструкции PROG_COMMANDS.
2. Разрешение записи ЭСППЗУ с помощью инструкции программирования 4a.
3. Загрузка старшего байта адреса с помощью инструкции программирования 4b.
4. Загрузка младшего байта адреса с помощью инструкции программирования 4c.
5. Загрузка данных с помощью инструкций 4d и 4e.
6. Повторение шагов 4 и 5 для всех байт данных в странице.
7. Запись данных с помощью инструкции программирования 4f.

8. Опрос завершения программирования ЭСППЗУ с помощью инструкции программирования 4g или задержка на время tWLRH (см. табл. 127).
9. Повторите шаги 3…8 до завершения программирования всех байт данных.

Обратите внимание, что инструкция PROG_PAGELOAD не может быть использована для программирования ЭСППЗУ.

Чтение ЭСППЗУ

1. Ввод JTAG-инструкции PROG_COMMANDS.
2. Разрешение чтения ЭСППЗУ с помощью инструкции программирования 5a.
3. Загрузка адреса с помощью инструкции программирования 5b и 5c.
4. Чтение данных с помощью инструкции программирования 5d.
5. Повторение шагов 3 и 4 до завершения считывания всех данных.

Обратите внимание, что инструкция PROG_PAGEREAD не может быть использована для чтения ЭСППЗУ.

Программирование конфигурационных бит

1. Ввод JTAG-инструкции PROG_COMMANDS.
2. Разрешение записи конфигурационных бит с помощью инструкции 6a.
3. Загрузка байта данных с помощью инструкции программирования 6b. Запись "0" приводит к программированию соответствующего конфигурационного бита, а запись "1" приводит к разпрограммированию конфигурационного бита.
4. Запись расширенного конфигурационного байта с помощью инструкции программирования 6c.
5. Опрос завершения записи конфигурационных бит с помощью инструкции программирования 6d или пауза tWLRH (см. табл. 127).
6. Загрузка байта данных с помощью инструкции программирования 6e. Запись "0" приводит к программированию соответствующего конфигурационного бита, а запись "1" приводит к разпрограммированию конфигурационного бита.
7. запись старшего конфигурационного байта с использованием инструкции программирования 6f.
8. Опрос завершения записи конфигурационных бит с помощью инструкции программирования 6g или пауза tWLRH (см. табл. 127).
9. Загрузка байта данных с помощью инструкции программирования 6h. Запись "0" приводит к программированию конфигурационного бита, а запись "1" приводит к разпрограммированию конфигурационного бита.
10. Запись младшего конфигурационного байта с помощью инструкции программирования 6i.
11. Опрос завершения записи конфигурационных бит с помощью инструкции программирования 6j или пауза tWLRH (см. табл. 127).

Программирование бит защиты

1. Ввод JTAG-инструкции PROG_COMMANDS.
2. Разрешение записи бита защиты с помощью инструкции программирования 7a.
3. Загрузка данных с помощью инструкции 7b. Загрузка "0" приводит к программирования бита защиты, а "1" оставляет его состояние неизменным.
4. Запись бит защиты с помощью инструкции программирования 7c.
5. Контроль завершения программирования бит защиты с помощью инструкции 7d или задержка на время tWLRH (см. табл.127).

Чтение конфигурационных бит и бит защиты

1. Ввод JTAG-инструкции PROG_COMMANDS.
2. Разрешение чтения конфигурационного бита/бита защиты, используя инструкцию 8a.
3. Для чтения всех конфигурационных бит и бит защиты используйте инструкцию программирования 8f.

Для чтения только расширенного конфигурационного байта используйте инструкцию программирования 8b.

Для чтения только старшего конфигурационного байта используйте инструкцию программирования 8c.

Для чтения только младшего конфигурационного байта используйте инструкцию программирования 8d.

Для чтения только бит защиты используйте инструкцию программирования 8e.

Чтение сигнатурных байт

1. Ввод JTAG-инструкции PROG_COMMANDS.
2. Разрешение чтения сигнатурного байта, используя инструкцию программирования 9a.
3. Загрузка адреса $00 с помощью инструкции программирования 9b.
4. Чтение первого сигнатурного байта при помощи инструкции программирования 9c.
5. Повторяем шаги 3 и 4 с адресами $01 и $02 для чтения 2-го и 3-го сигнатурного байта, соответственно.

Чтение калибровочного байта

1. Ввод JTAG-инструкции PROG_COMMANDS.
2. Разрешение чтения калибровочного байта, используя инструкцию программирования 10a.
3. Загрузка адреса $00, используя инструкцию программирования 10b.
4. Чтение калибровочного байта, используя инструкцию программирования 10c.

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.изм.
V_{IL}	Входное напряжение низкого уровня	Кроме выводов XTAL1 и RESET	-0.5		0.2 V_{CC} (1)	В
V_{IL1}	Входное напряжение низкого уровня	вывод XTAL1, выбрана внешняя синхронизация	-0.5		0.1 V_{CC} (1)	В

V_{IL2}	Входное напряжение низкого уровня	вывод сброса RESET	-0.5		0.2 V_{CC} (1)	В
V_{IH}	Входное напряжение высокого уровня	Кроме выводов XTAL1 , RESET	0.6 V_{CC} (2)		VCC + 0.5	В
V_{IH1}	Входное напряжение высокого уровня	Вывод XTAL1, выбрана внешняя синхронизация	0.7 V_{CC} (2)		V_{CC} + 0.5	В
V_{IH2}	Входное напряжение высокого уровня	Вывод сброса RESET	0.85 V_{CC} (2)		V_{CC} + 0.5	В
V_{OL}	Выходное напряжение низкого уровня (3)(порты A,B,C,D, E, F, G)	I_{OL} = 20 мА, V_{CC} = 5В I_{OL} = 10 мА, V_{CC} = 3В			0.7 0.5	В
V_{OH}	Выходное напряжение высокого уровня (4)(порты A,B,C,D)	I_{OH} = -20 мА, V_{CC} = 5В I_{OH} = -10 мА, V_{CC} = 3В	4.0 2.2			В
I_{IL}	Входной ток утечки через линию ввода-вывода	V_{cc} = 5.5В, лог. 0 (абс. значение)			8.0	мкА
I_{IH}	Входной ток утечки через линию ввода-вывода	V_{cc} = 5.5В, лог. 1 (абс. значение)			8.0	мкА
R_{RST}	Сопротивление подтягивающего резистора на входе сброса		30		100	кОм
R_{PEN}	Сопротивление подтягивающего резистора на входе PEN		25		100	кОм
R_{PU}	Сопротивление подтягивающего резистора на линиях ввода-вывода		20		100	кОм
I_{CC}	Потребляемый ток	4 МГц, V_{CC} = 3В, активный режим (ATmega128L)			5	мА
		8 МГц, VCC = 5В, активный режим (ATmega128)			20	мА
		4 МГц, V_{CC} = 3В, режим холостого хода (ATmega128L)			2	мА
		8 МГц, V_{CC} = 5В, режим холостого хода (ATmega128)			12	мА
	Режим выключения (Power-down)(5)	Стор. таймер включен, V_{CC} = 3В		< 25	40	мкА
		Стор. таймер отключен, V_{CC} = 3В		< 10	25	мкА
V_{ACIO}	Входное напряжение смещения аналогового компаратора	V_{CC} = 5В $V_{вх}$ = V_{CC}/2			40	мВ
I_{ACLK}	Входной ток утечки аналогового компаратора	V_{CC} = 5В $V_{вх}$ = V_{CC}/2	-50		50	нА
t_{ACID}	Задержка на инициализацию аналогового компаратора	V_{CC} = 2.7В V_{CC} = 5.0В	750 500			нс
t_{ACID}	Задержка распространению сигнала в аналоговом компараторе	V_{CC} = 2.7В V_{CC} = 5.0В	750500			нс

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 TQFP и MLF:
1] Суммарные токи IOL всех линий ввода-вывода не должны превышать 400 мА.
2] Суммарные токи IOL всех линий ввода-вывода A0 - A7, G2, C3 - C7 не должны превышать 300 мА.
3] Суммарные токи IOL всех линий ввода-вывода C0 - C2, G0 - G1, D0 - D7, XTAL2 не должны превышать 150 мА.
4] Суммарные токи IOL всех линий ввода-вывода B0 - B7, G3 - G4, E0 - E7 не должны превышать 150 мА.
5] Суммарные токи IOL всех линий ввода-вывода F0 - F7 не должны превышать 200 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:

   Корпус TQFP и MLF:
   1] Суммарные токи IOH всех линий ввода-вывода не должны превышать 400 мА.
   2] Суммарные токи IOH всех линий ввода-вывода A0 - A7, G2, C3 - C7 не должны превышать 300 мА.
   3] Суммарные токи IOH всех линий ввода-вывода C0 - C2, G0 - G1, D0 - D7, XTAL2 не должны превышать 150 мА.
   4] Суммарные токи IOH всех линий ввода-вывода B0 - B7, G3 - G4, E0 - E7 не должны превышать 150 мА.
   5] Суммарные токи IOH всех линий ввода-вывода F0 - F7 не должны превышать 200 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

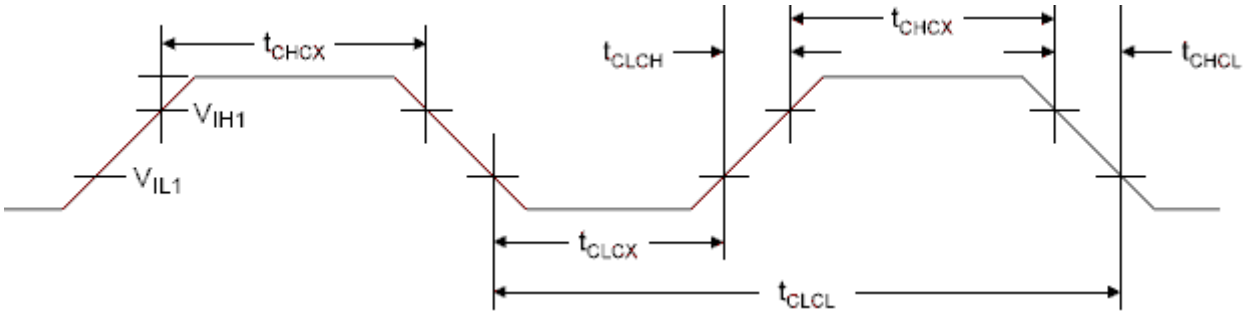


Рисунок 152. Осциллограмма внешнего тактового сигнала

Таблица 132. 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 133. Типичные частоты при тактировании от внешней RC-цепи

R, кОм	C, пФ	f
100	70	$TBD_{(2)}$
31,5	20	$TBD_{(2)}$
6,5	20	$TBD_{(2)}$

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. TBD означает, что точное значение величины находится в состоянии определения.

Характеристики двухпроводного последовательного интерфейса

В таблице 134 описываются требования к устройствам, подключаемых к двухпроводной последовательной шине. Двухпроводной последовательный интерфейс ATmega128 отвечает данным требованиям или превосходит их в указанных условиях.

Обозначения параметров показаны на рисунке 153.

Таблица 134. 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 ATmega128 проверено не полностью.
2. Только для f_{SCL} > 100 кГц
3. Cb - емкость одной линии шины в пФ.
4. f_{CK} - тактовая частота ЦПУ
5. Данное требования относится ко всей работе двухпроводного последовательного интерфейса ATmega128. 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 ($1/f_{SCL}$ - $2/f_{CK}$), таким образом, частота f_{CK} должна быть больше 6 МГц для более точного выполнения требования к длительности низкого уровня при f_{SCL} = 100 кГц.
7. Фактическая длительность низкого уровня, генерируемого двухпроводным последовательным интерфейсом ATmega128, составляет ($1/f_{SCL}$ - $2/f_{CK}$), таким образом, требования к длительности низкого уровня не строго выполняются для f_{SCL} > 308 кГц, когда f_{CK} = 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 t_{LOW}.

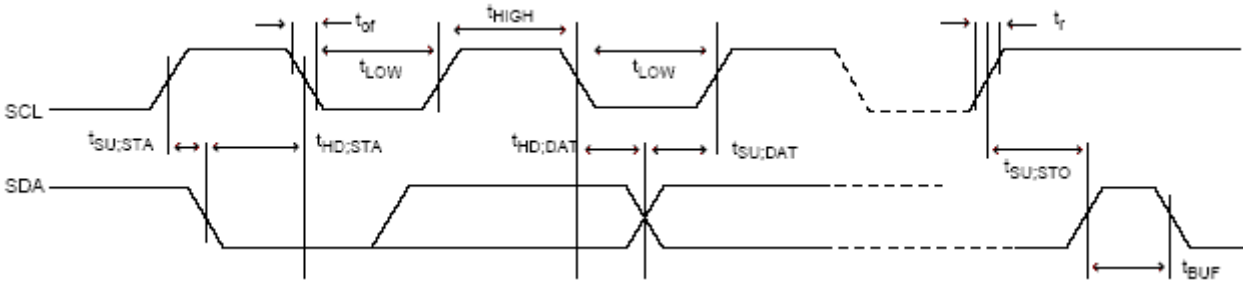


Рисунок 153. Временная диаграмма двухпроводной последовательной шины

Характеристики временной диаграммы SPI

Таблица 135. Параметры временной диаграммы SPI

№ пп	Описание	Режим	Мин.	Тип.	Макс.	Ед.изм.
1	Период SCK	ведущий		см. табл. 72		нс
2	Длительность низкого/высокого уровня SCK	ведущий		меандр		
3	Время нарастания/ спада	ведущий		TBD (2)		
4	Установка	ведущий		10		
5	Удержание	ведущий		10		
6	Вывод к SCK	ведущий		0.5 · t_{sck}		

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	$4 \cdot t_{ck}$			
11	Длительность низкого/высокого уровня SCK[(1)]	подчиненный	$4 \cdot t_{ck}$			
12	Время нарастания/ спада	подчиненный		TBD		
13	Установка	подчиненный	10			
14	Удержание	подчиненный	t_{ck}			
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	20			

Прим.:

1. В режиме программирования через SPI минимальные длительности низкого/высокого уровней SCK должны быть следующие:

   - 2 t_{CLCL} для f_{CK} < 12 МГц
   - 3 t_{CLCL} для f_{CK} > 12 МГц

2. TBD означает, что точное значение величины находится в состоянии определения.

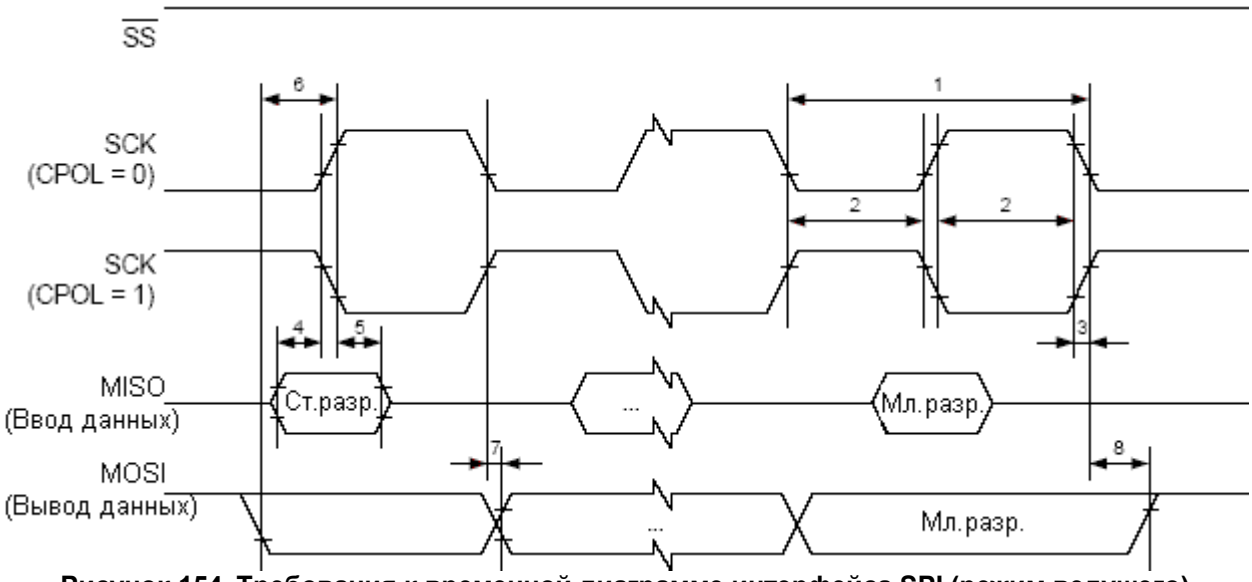


Рисунок 154. Требования к временной диаграмме интерфейса SPI (режим ведущего)

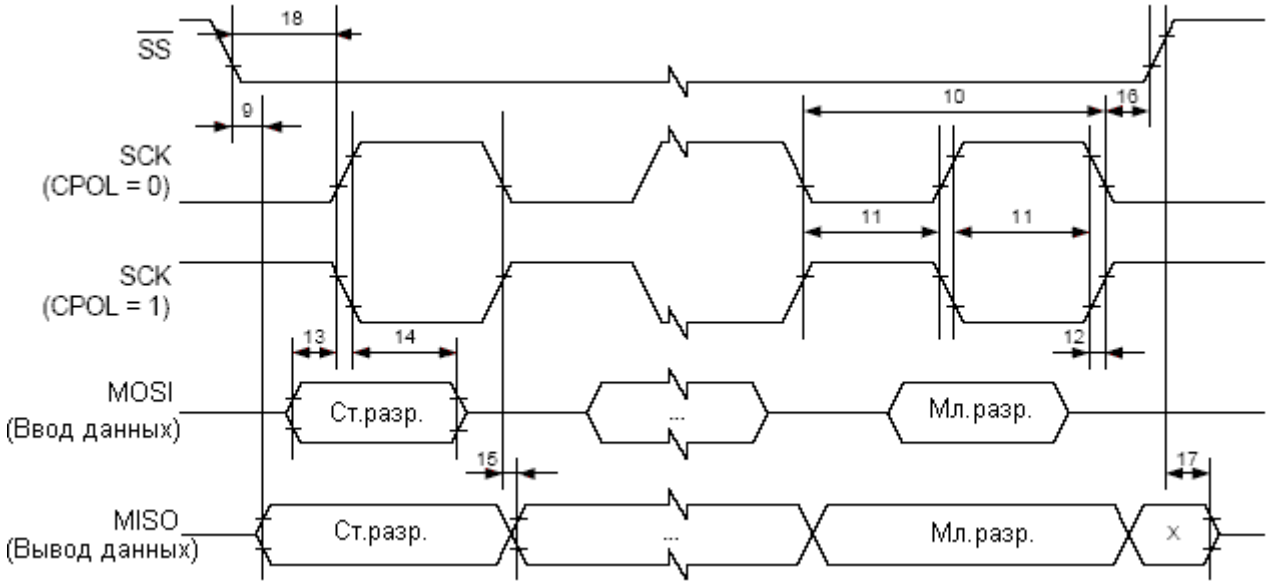


Рисунок 155. Требования к временной диаграмме интерфейса SPI (режим подчиненного)

Предварительные данные по характеристикам АЦП

Таблица 136. Характеристики АЦП

Обозн.	Параметр	Условия измерения	Мин.(1)	Тип.(1)	Макс.(1)	Ед.изм.
	Разрешающая способность	Одиночное преобразование		10		бит
		Дифференциальное преобразование с усилением 1x или 20x		8		бит
		Дифференциальное преобразование с усилением 200x		7		бит
	Абсолютная погрешность	Одиночное преобразование; $V_{ион}$=4В $F_{синхр.ацп}$=200кГц		1	TBD (4)	мл.разр.
	Интегральная нелинейность	$V_{ион}$=4В		0.5		мл.разр.
	Дифференциальная нелинейность	Vион=4В		0.5		мл.разр.
	Погрешность в начале шкалы	Vион=4В		1		мл.разр.
	Время преобразования	Автоматическое преобразование	65		250	мкс
	Тактовая частота		50		200	кГц
AV_{CC}	Аналоговое напряжение питания		V_{CC} - 0.3(2)		V_{CC} + 0.3(3)	В
$V_{ион}$	Опорное напряжение	Одиночное преобразование	2.0		AV_{CC}	В
		Дифференциальное преобразование	2.0		AV_{CC}- 0.2	В
Vвх	Входное напряжение	Одиночное преобразование	GND		Vион	
		Дифференциальное преобразование	TBD(4)		TBD(4)	
	Входной частотный диапазон	Одиночное преобразование				
		Дифференциальное				

		преобразование				
Vвн.ион	Напряжение внутреннего опорного источника		2.4	2.56	2.8	В
Rион	Сопротивление входа подключения опорного источника		TBD(4)	TBD(4)	TBD(4)	кОм
Rвх	Сопротивление аналогового входа			TBD(4)		кОм

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. TBD означает, что точное значение величины находится в состоянии определения.

Временная диаграмма внешней памяти данных

Таблица 137. Характеристики внешней памяти данных (4.5 - 5.5В, без состояний ожидания)

	Обозначение	Параметр	Генератор 8МГц		Переменный генератор		Ед.изм.
			мин.	макс.	мин.	макс.	
0	$1/t_{CLCL}$	Частота генератора			0.0	16	МГц
1	t_{LHLL}	Длительность импульса ALE	115		$1.0t_{CLCL}$-10		нс
2	t_{AVLL}	Действительность адреса A до низкого уровня ALE	57.5		$0.5t_{CLCL}$-5(1)		нс
3a	t_{LLAX_ST}	Удержание адреса после установки низкого уровня на ALE во время записи	5		5		нс
3b	t_{LLAX_LD}	Удержание адреса после установки низкого уровня на ALE во время чтения	5		5		нс
4	t_{AVLLC}	Действительность адреса C до низкого уровня ALE	57,5		$0.5t_{CLCL}$-5(1)		нс
5	t_{AVRL}	Действительность адреса до низкого уровня на RD	115		$1.0t_{CLCL}$-10		нс
6	t_{AVWL}	Действительность адреса до низкого уровня на WR	115		$1.0t_{CLCL}$-10		нс
7	t_{LLWL}	Время после установки низкого уровня на ALE до появления низкого уровня на WR	47,5	67,5	$0.5t_{CLCL}$-15(2)	$0.5t_{CLCL}$+5(2)	нс
8	t_{LLRL}	Время после установки низкого уровня на ALE до появления низкого уровня на RD	47,5	67,5	$0.5t_{CLCL}$-15(2)	$0.5t_{CLCL}$+5(2)	нс
9	t_{DVRH}	Готовность данных до появления высокого уровня на RD	40		40		нс
10	t_{RLDV}	Время подготовки данных после установки лог. 0 на RD		75		$1.0t_{CLCL}$-50	нс
11	t_{RHDX}	Удержание данных после установки лог. 1 на RD	0		0		нс
12	t_{RLRH}	Длительность импульса чтения RD	115		$1.0t_{CLCL}$-10		нс
13	t_{DVWL}	Врем готовности данных до появления лог. 0 на WR	42,5		$0.5t_{CLCL}$-20(1)		нс

14	t_{WHDX}	Удержание данных после подачи лог. 1 на WR	115		$1.0t_{CLCL}$-10		нс
15	t_{DVWH}	Действительность данных до появления лог.1 на WR	125		$1.0t_{CLCL}$		нс
16	t_{WLWH}	Длительность импульса записи WR	115		$1.0t_{CLCL}$-10		нс

Прим.:

1. Здесь полагается 50%-ое заполнение синхронизирующих импульсов. Половина периода фактически равна длительности единичного импульса внешнего тактового сигнала на XTAL1.
2. Здесь полагается 50%-ое заполнение синхронизирующих импульсов. Половина периода фактически равна длительности нулевого импульса внешнего тактового сигнала на XTAL1.

Таблица 138. Характеристики внешней памяти данных с однотактным состоянием ожидания (4.5 - 5.5В)

	Обозначение	Параметр	Генератор 8МГц		Переменный генератор		Ед.изм.
			мин.	макс.	мин.	макс.	
0	$1/t_{CLCL}$	Частота генератора			0.0	16	МГц
10	t_{RLDV}	Время подготовки данных после установки лог. 0 на RD		200		$2.0t_{CLCL}$-50	нс
12	t_{RLRH}	Длительность импульса чтения RD	240		$2.0t_{CLCL}$-10		нс
15	t_{DVWH}	Действительность данных до появления лог.1 на WR	240		$2.0t_{CLCL}$		нс
16	t_{WLWH}	Длительность импульса записи WR	240		$1.0t_{CLCL}$-10		нс

Таблица 139. Характеристики внешней памяти данных (4.5 - 5.5В, SRWn1 = 1, SRWn0 = 0)

	Обозначение	Параметр	Генератор 8МГц		Переменный генератор		Ед.изм.
			мин.	макс.	мин.	макс.	
0	$1/t_{CLCL}$	Частота генератора			0.0	16	МГц
10	t_{RLDV}	Время подготовки данных после установки лог. 0 на RD		325		$3.0t_{CLCL}$-50	нс
12	t_{RLRH}	Длительность импульса чтения RD	365		$3.0tC_{LCL}$-10		нс
15	t_{DVWH}	Действительность данных до появления лог.1 на WR	375		$3.0t_{CLCL}$		нс
16	t_{WLWH}	Длительность импульса записи WR	365		$3.0t_{CLCL}$-10		нс

Таблица 140. Характеристики внешней памяти данных (4.5 - 5.5В, SRWn1 = 1, SRWn0 = 1)

	Обозначение	Параметр	Генератор 8МГц		Переменный генератор		Ед.изм.
			мин.	макс.	мин.	макс.	
0	$1/t_{CLCL}$	Частота генератора			0.0	16	МГц
10	t_{RLDV}	Время подготовки данных после установки лог. 0 на RD		325		$3.0t_{CLCL}$-50	нс
12	t_{RLRH}	Длительность импульса чтения RD	365		$3.0t_{CLCL}$-10		нс

14	t_{WHDX}	Удержание данных после подачи лог. 1 на WR	240		$2.0t_{CLCL}$-10		нс
15	t_{DVWH}	Действительность данных до появления лог.1 на WR	375		$3.0t_{CLCL}$		нс
16	t_{WLWH}	Длительность импульса записи WR	365		$3.0t_{CLCL}$-10		нс

Таблица 141. Характеристики внешней памяти данных (2.7 - 5.5В, без состояний ожидания)

	Обозначение	Параметр	Генератор 4МГц		Переменный генератор		Ед.изм.
			мин.	макс.	мин.	макс.	
0	$1/t_{CLCL}$	Частота генератора			0.0	8	МГц
1	t_{LHLL}	Длительность импульса ALE	235		$1.0t_{CLCL}$-15		нс
2	t_{AVLL}	Действительность адреса A до низкого уровня ALE	115		$0.5t_{CLCL}$-10(1)		нс
3a	t_{LLAX_ST}	Удержание адреса после установки низкого уровня на ALE во время записи	5		5		нс
3b	t_{LLAX_LD}	Удержание адреса после установки низкого уровня на ALE во время чтения	5		5		нс
4	t_{AVLLC}	Действительность адреса C до низкого уровня ALE	115		$0.5t_{CLCL}$-10(1)		нс
5	t_{AVRL}	Действительность адреса до низкого уровня на RD	235		$1.0t_{CLCL}$-15		нс
6	t_{AVWL}	Действительность адреса до низкого уровня на WR	235		$1.0t_{CLCL}$-15		нс
7	t_{LLWL}	Время после установки низкого уровня на ALE до появления низкого уровня на WR	115	130	$0.5t_{CLCL}$-10(2)	$0.5t_{CLCL}$+5(2)	нс
8	t_{LLRL}	Время после установки низкого уровня на ALE до появления низкого уровня на RD	115	130	$0.5t_{CLCL}$-10(2)	$0.5t_{CLCL}$+5(2)	нс
9	t_{DVRH}	Готовность данных до появления высокого уровня на RD	45		45		нс
10	t_{RLDV}	Время подготовки данных после установки лог. 0 на RD		190		$1.0t_{CLCL}$-50	нс
11	t_{RHDX}	Удержание данных после установки лог. 1 на RD	0		0		нс
12	t_{RLRH}	Длительность импульса чтения RD	235		$1.0t_{CLCL}$-15		нс
13	t_{DVWL}	Время готовности данных до появления лог. 0 на WR	105		$0.5t_{CLCL}$-20(1)		нс
14	t_{WHDX}	Удержание данных после подачи лог. 1 на WR	235		$1.0t_{CLCL}$-15		нс
15	t_{DVWH}	Действительность данных до появления лог.1 на WR	250		$1.0t_{CLCL}$		нс
16	t_{WLWH}	Длительность импульса записи WR	235		$1.0t_{CLCL}$-15		нс

Прим.:

1. Здесь полагается 50%-ое заполнение синхронизирующих импульсов. Половина периода фактически равна длительности единичного импульса внешнего тактового сигнала на XTAL1.

2. Здесь полагается 50%-ое заполнение синхронизирующих импульсов. Половина периода фактически равна длительности нулевого импульса внешнего тактового сигнала на XTAL1.

Таблица 142. Характеристики внешней памяти данных (2.7 - 5.5В, SRWn1 = 0, SRWn0 = 1)

	Обозначение	Параметр	Генератор 4МГц		Переменный генератор		Ед.изм.
			мин.	макс.	мин.	макс.	
0	$1/t_{CLCL}$	Частота генератора			0.0	8	МГц
10	t_{RLDV}	Время подготовки данных после установки лог. 0 на RD		440		$2.0t_{CLCL}$-60	нс
12	t_{RLRH}	Длительность импульса чтения RD	485		$2.0t_{CLCL}$-15		нс
15	t_{DVWH}	Действительность данных до появления лог.1 на WR	500		$2.0t_{CLCL}$		нс
16	t_{WLWH}	Длительность импульса записи WR	485		$2.0t_{CLCL}$-15		нс

Таблица 143. Характеристики внешней памяти данных (2.7 - 5.5В, SRWn1 = 1, SRWn0 = 0)

	Обозначение	Параметр	Генератор 4МГц		Переменный генератор		Ед.изм.
			мин.	макс.	мин.	макс.	
0	$1/t_{CLCL}$	Частота генератора			0.0	8	МГц
10	t_{RLDV}	Время подготовки данных после установки лог. 0 на RD		690		$3.0t_{CLCL}$-60	нс
12	t_{RLRH}	Длительность импульса чтения RD	735		$3.0t_{CLCL}$-15		нс
15	t_{DVWH}	Действительность данных до появления лог.1 на WR	750		$3.0t_{CLCL}$		нс
16	t_{WLWH}	Длительность импульса записи WR	735		$3.0t_{CLCL}$-15		нс

Таблица 144. Характеристики внешней памяти данных (2.7 - 5.5В, SRWn1 = 1, SRWn0 = 1)

	Обозначение	Параметр	Генератор 4МГц		Переменный генератор		Ед.изм.
			мин.	макс.	мин.	макс.	
0	$1/t_{CLCL}$	Частота генератора			0.0	8	МГц
10	t_{RLDV}	Время подготовки данных после установки лог. 0 на RD		690		$3.0t_{CLCL}$-60	нс
12	t_{RLRH}	Длительность импульса чтения RD	735		$3.0t_{CLCL}$-15		нс
14	t_{WHDX}	Удержание данных после подачи лог. 1 на WR	485		$2.0t_{CLCL}$-15		нс
15	t_{DVWH}	Действительность данных до появления лог.1 на WR	750		$3.0t_{CLCL}$		нс
16	t_{WLWH}	Длительность импульса записи WR	735		$3.0t_{CLCL}$-15		нс

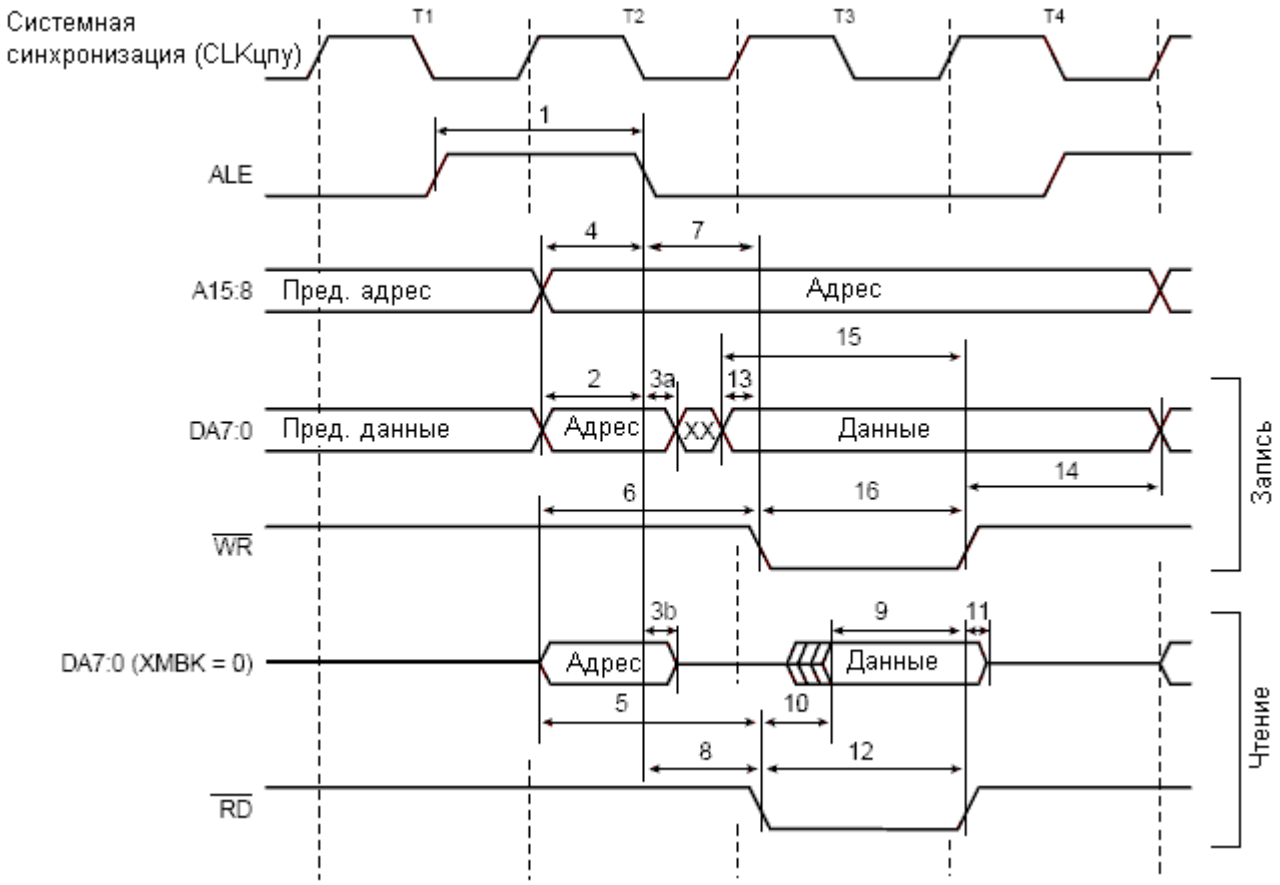


Рисунок 156. Временная диаграмма внешней памяти (SRWn1 = 0, SRWn0 = 0)

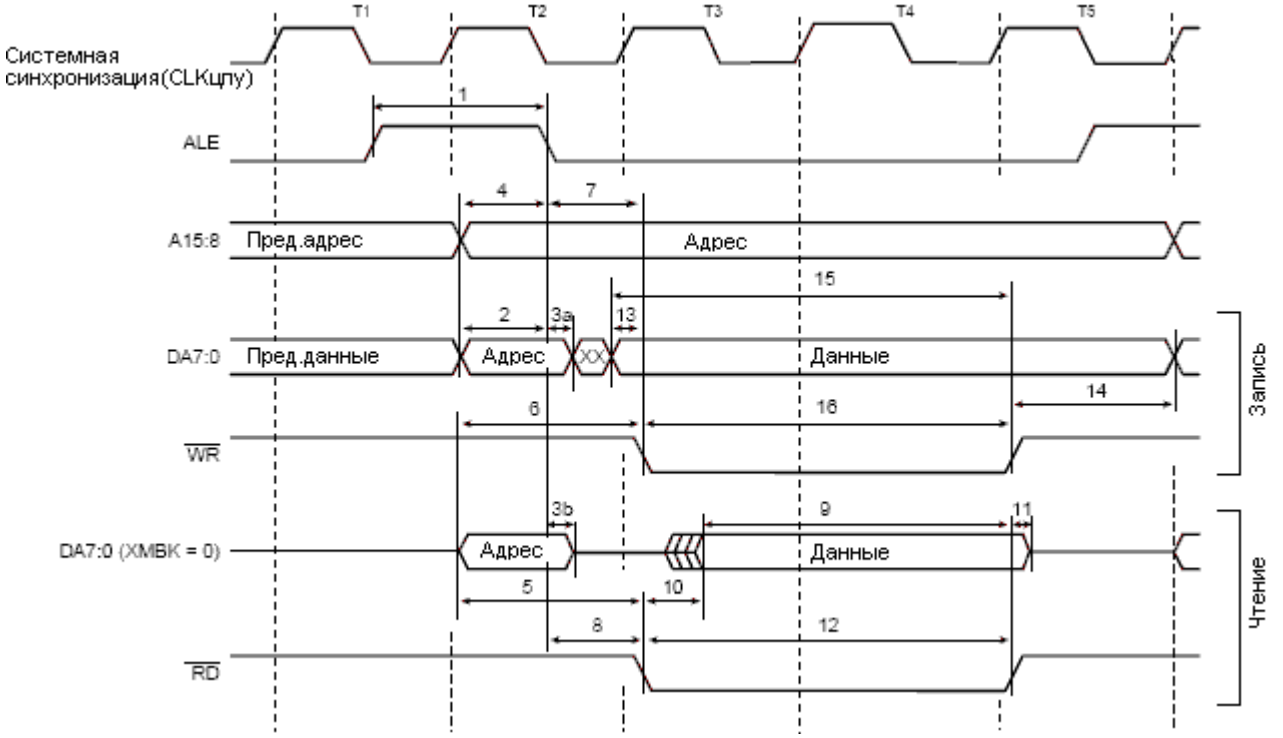


Рисунок 157. Временная диаграмма внешней памяти (SRWn1 = 0, SRWn0 = 1)

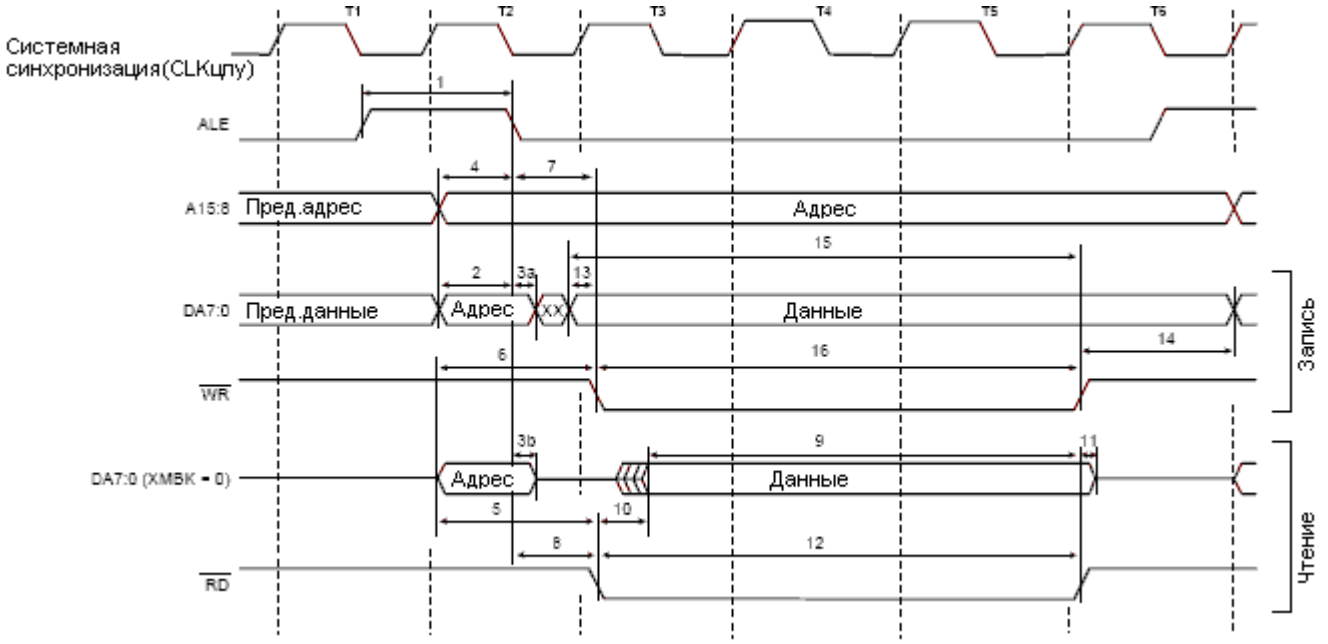


Рисунок 158. Временная диаграмма внешней памяти (SRWn1 = 1, SRWn0 = 0)

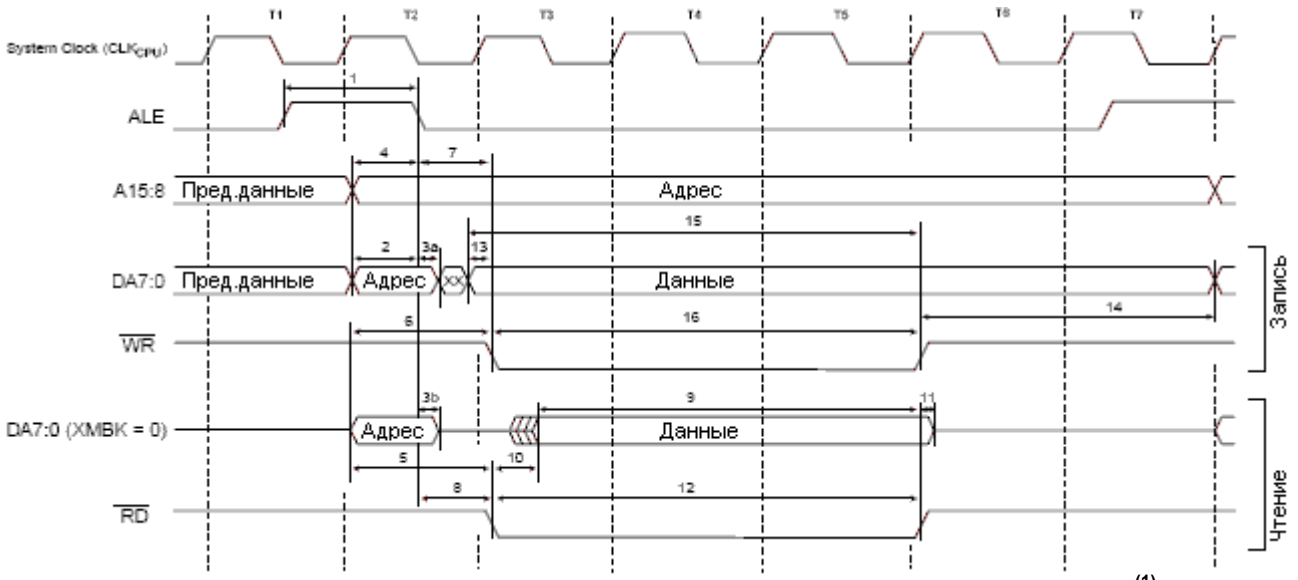


Рисунок 158. Временная диаграмма внешней памяти (SRWn1 = 1, SRWn0 = 1)[1]

Прим.:

1. В последнем периоде (T4-T7) импульс ALE присутствует только в том случае, если следующая инструкция осуществляет доступ к ОЗУ (внутреннему или внешнему).

Типовые характеристики ATmega128: предварительные данные

На следующих рисунках приведены типичные характеристики AMmega128. В процессе производства соответствие данным характеристикам не проверяется. Измерение всех характеристик энергопотребления выполнены при конфигурации всех портов ввода-вывода на ввод и активизации подтягивающих резисторов. В качестве источника синхронизации использовался синусоидальный генератор с выходом, изменяющимся между уровнями питания (rail-to-rail).

Потребление в режиме выключения (Power-down) не зависит от выбора источника синхронизации.

Потребляемый ток зависит от нескольких факторов: напряжение питания, рабочая частота, нагрузка линий ввода-вывода, частота переключения линий ввода-вывода, выполняемый код и окружающая температура. Доминирующими факторами являются рабочее напряжение и частота.

Если вывод нагружен емкостной нагрузкой, то протекаемый ток может быть вычислен как $C_L*V_{CC}*f$, где C_L - емкость нагрузки, V_{CC} - рабочее напряжения питания и f - средняя частота переключения линии ввода-вывода.

Не гарантируется нормальная работа узлов микроконтроллера на частотах выше указанной в коде заказа Отличие в потребляемом токе режима выключения с включенным сторожевым таймером и режима выключения с выключенным сторожевым таймером состоит в потребляемом токе сторожевого таймера.

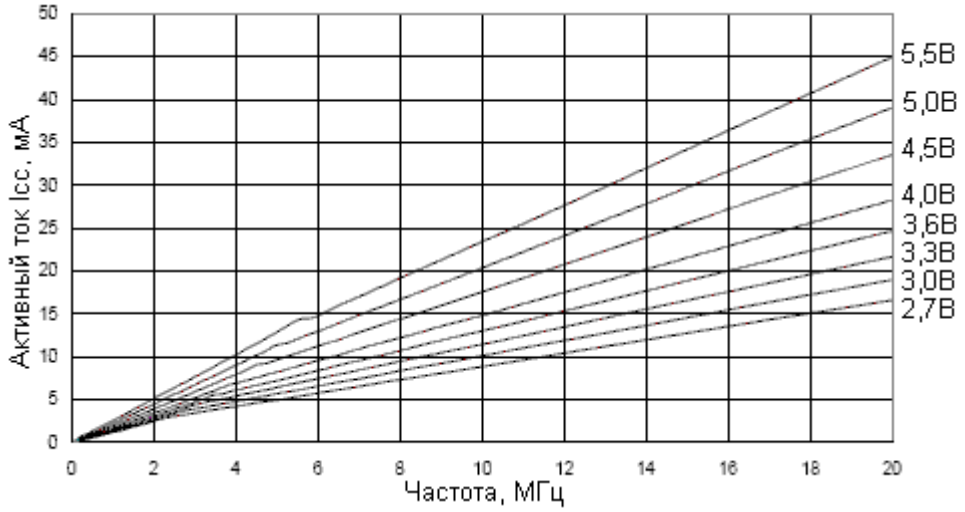


Рисунок 160. Зависимость активного потребляемого тока от тактовой частоты

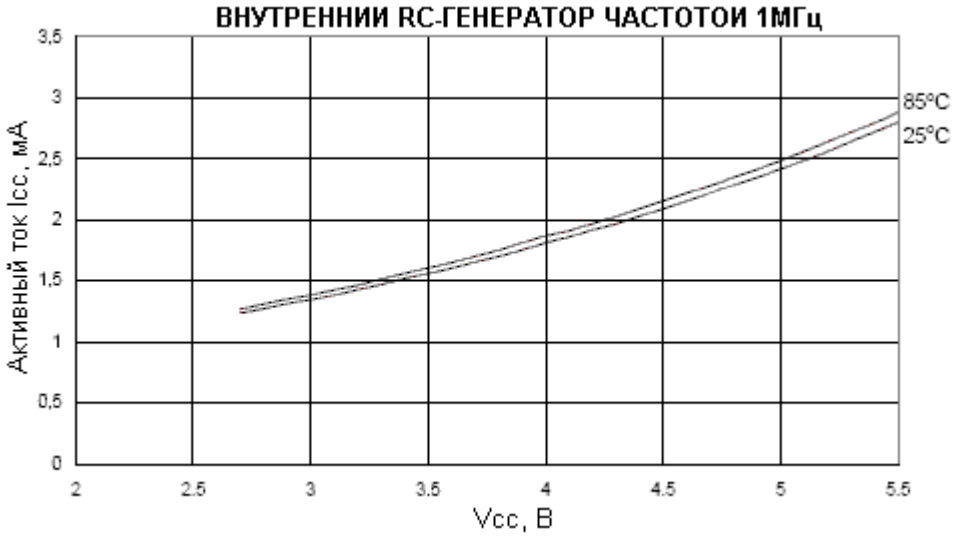


Рисунок 161. Зависимость активного потребляемого тока от напряжения питания (тактирование внутренним RC-генератором частотой 1МГц)

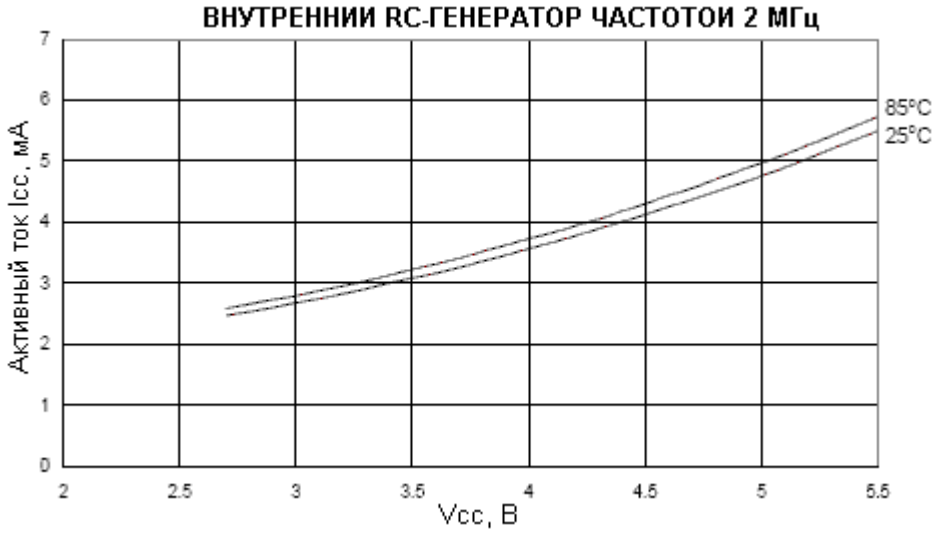


Рисунок 162. Зависимость активного потребляемого тока от напряжения питания (тактирование внутренним RC-генератором частотой 2МГц)

Рисунок 163. Зависимость активного потребляемого тока от напряжения питания (тактирование внутренним RC-генератором частотой 4МГц)

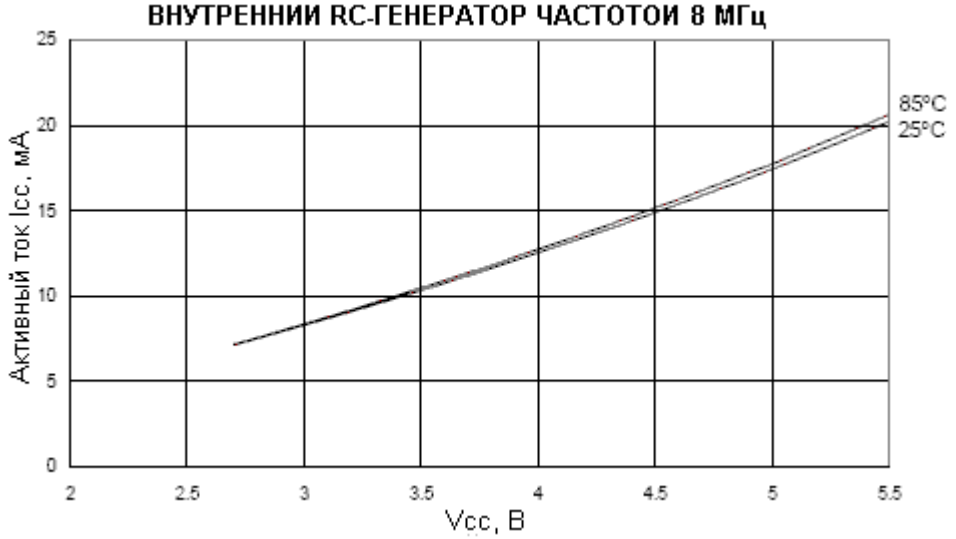


Рисунок 164. Зависимость активного потребляемого тока от напряжения питания (тактирование внутренним RC-генератором частотой 8МГц)

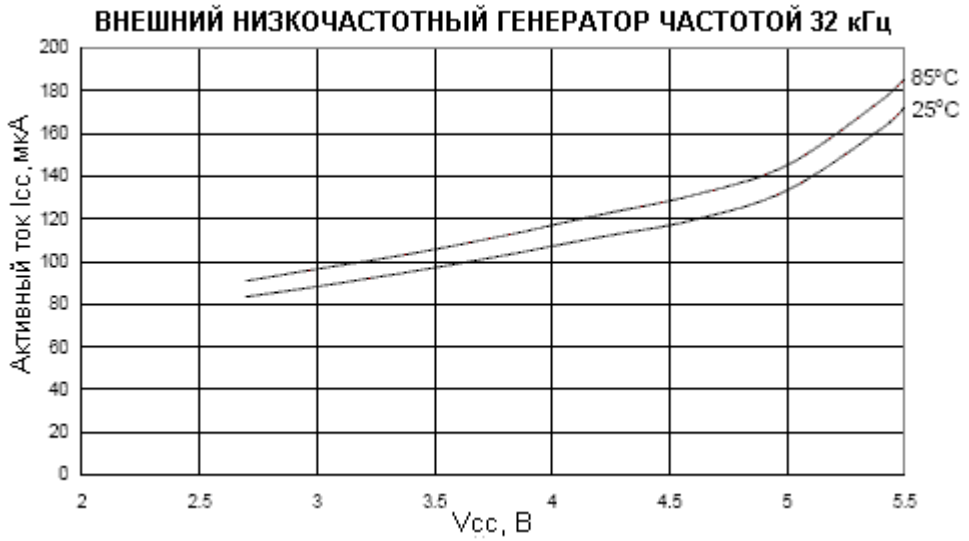


Рисунок 165. Зависимость активного потребляемого тока от напряжения питания (тактирование внешним низкочастотным генератором частотой 32 кГц)

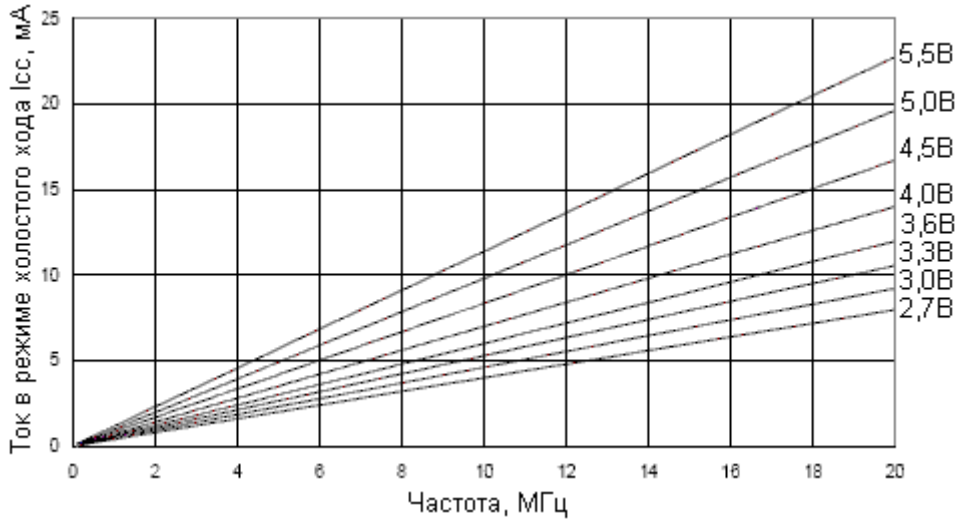


Рисунок 166. Зависимость потребляемого тока в режиме холостого хода (Idle) от тактовой частоты

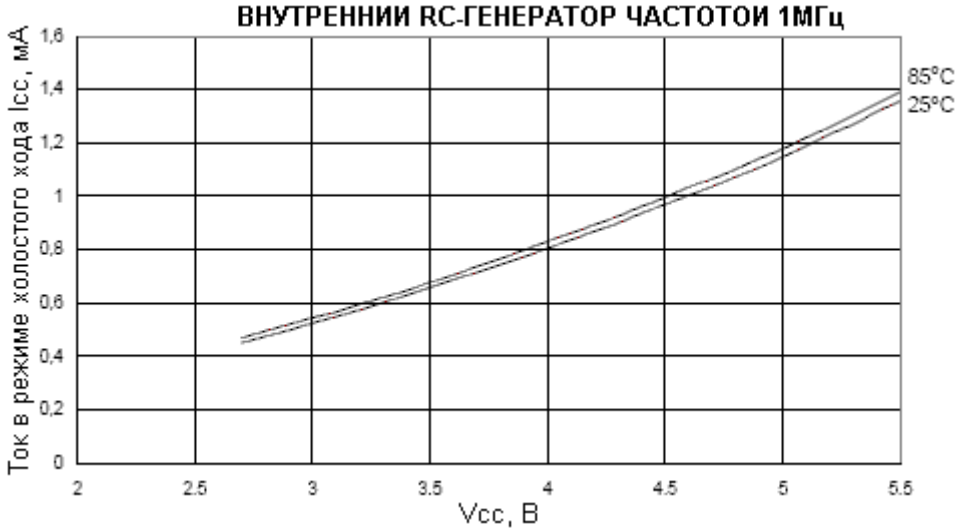


Рисунок 167. Зависимость потребляемого тока в режиме холостого хода (Idle) от напряжения питания (тактирование внутренним RC-генератором частотой 1МГц)

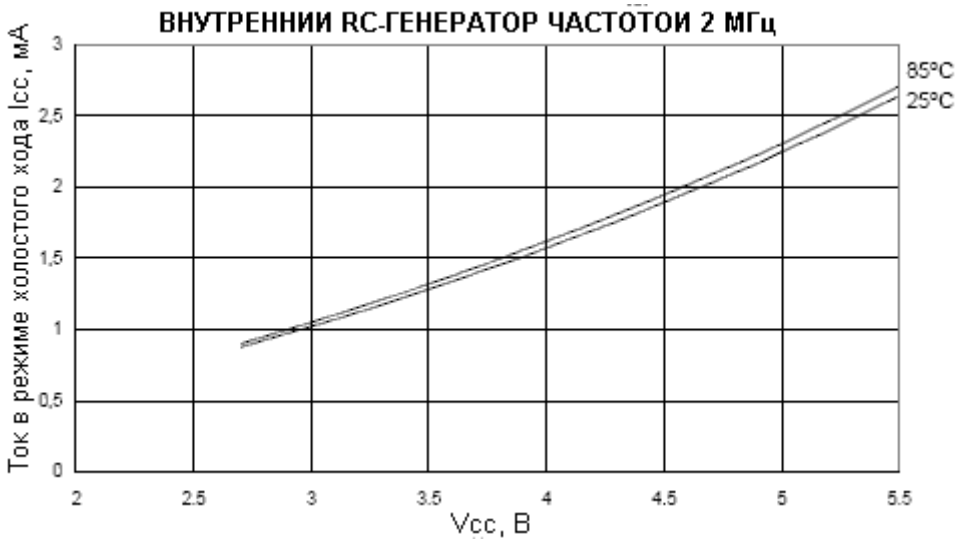


Рисунок 168. Зависимость потребляемого тока в режиме холостого хода (Idle) от напряжения питания (тактирование внутренним RC-генератором частотой 2МГц)

Рисунок 169. Зависимость потребляемого тока в режиме холостого хода (Idle) от напряжения питания (тактирование внутренним RC-генератором частотой 4МГц)

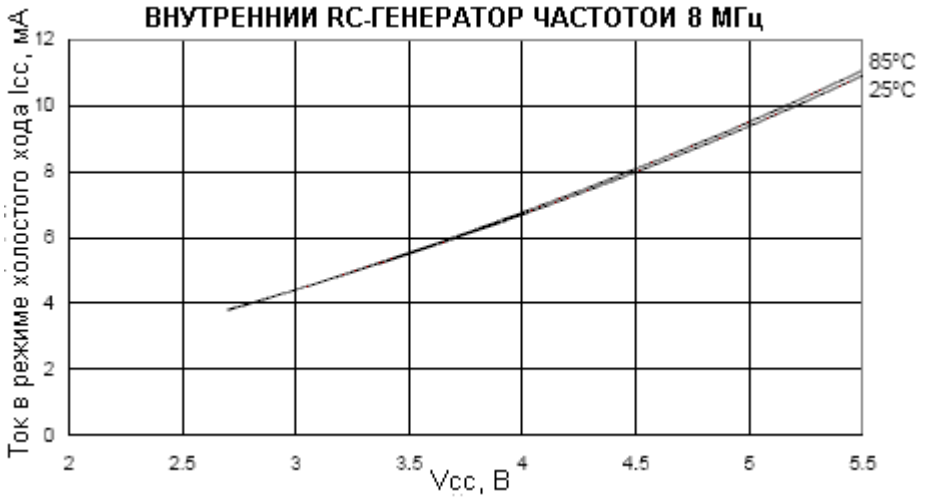


Рисунок 170. Зависимость потребляемого тока в режиме холостого хода (Idle) от напряжения питания (тактирование внутренним RC-генератором частотой 8МГц)

Рисунок 171. Зависимость потребляемого тока в режиме холостого хода (Idle) от напряжения питания (тактирование внешним низкочастотным генератором частотой 32 кГц)

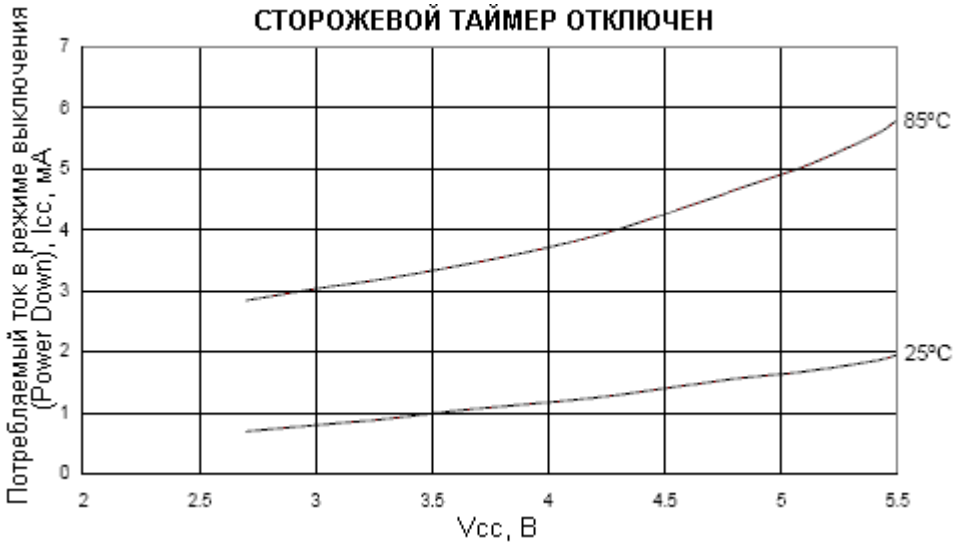


Рисунок 172. Зависимость потребляемого тока в режиме выключения (Power-down) от напряжения питания (сторожевой таймер отключен)

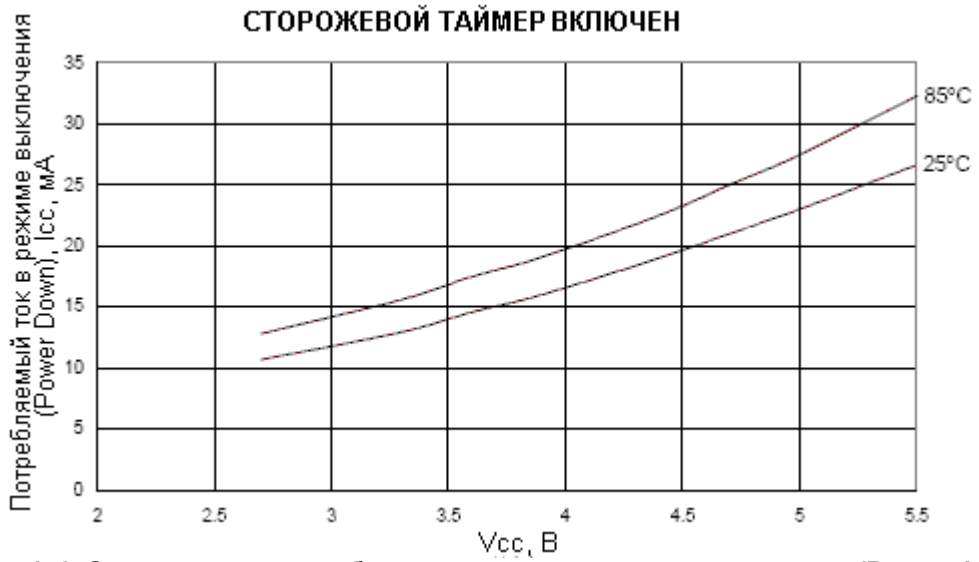


Рисунок 173. Зависимость потребляемого тока в режиме выключения (Power-down) от напряжения питания (сторожевой таймер включен)

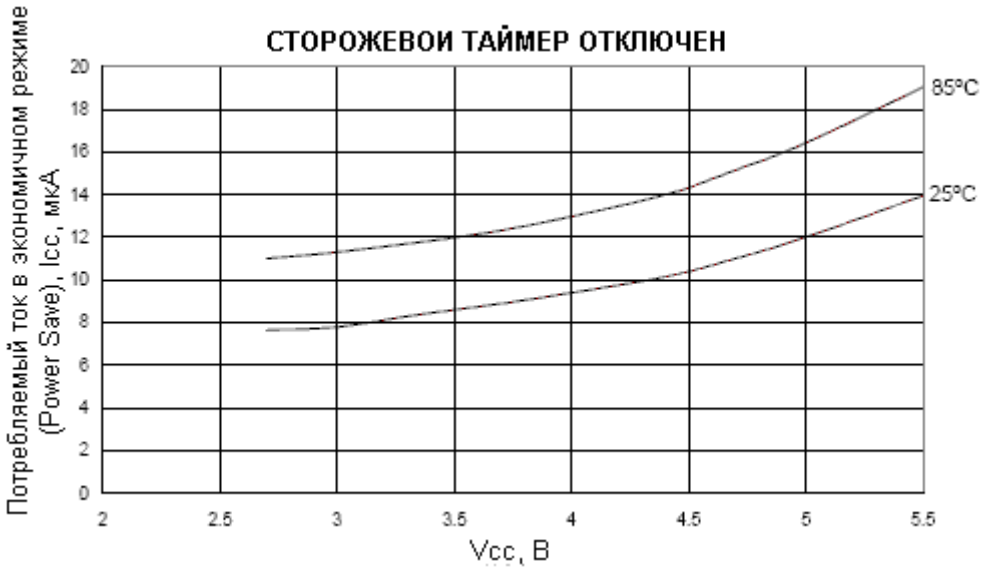


Рисунок 174. Зависимость потребляемого тока в экономичном режиме (Power-save) от напряжения питания (сторожевой таймер отключен)

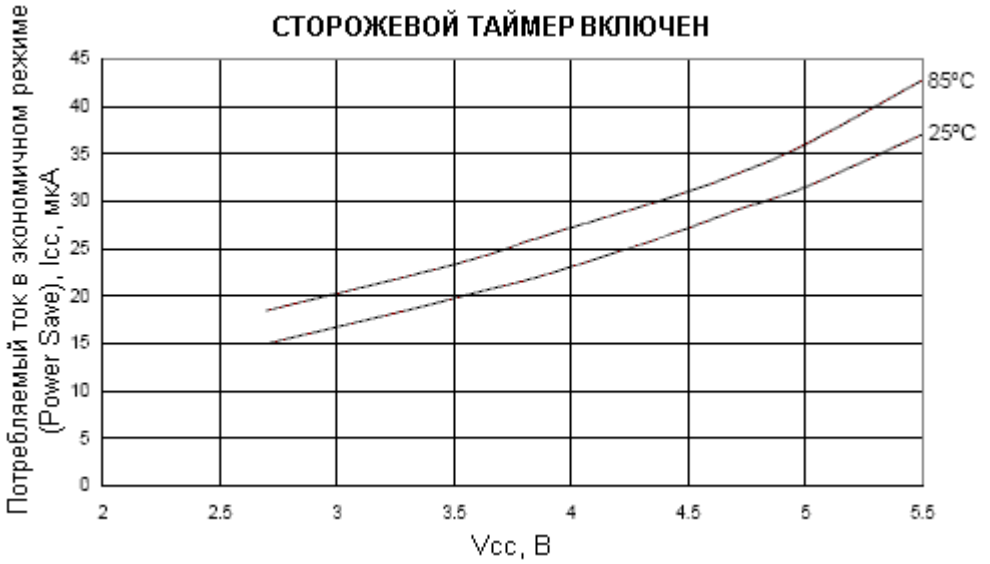


Рисунок 175. Зависимость потребляемого тока в экономичном режиме (Power-save) от напряжения питания (сторожевой таймер включен)

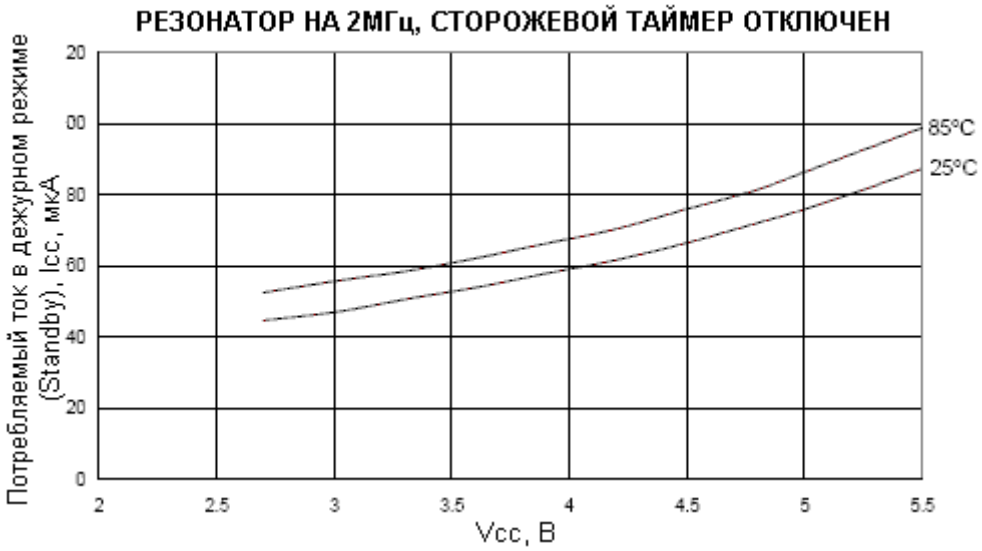


Рисунок 176. Зависимость потребляемого тока в дежурном режиме (Standby) от напряжения питания (тактирование резонатором частотой 2МГц, сторожевой таймер отключен)

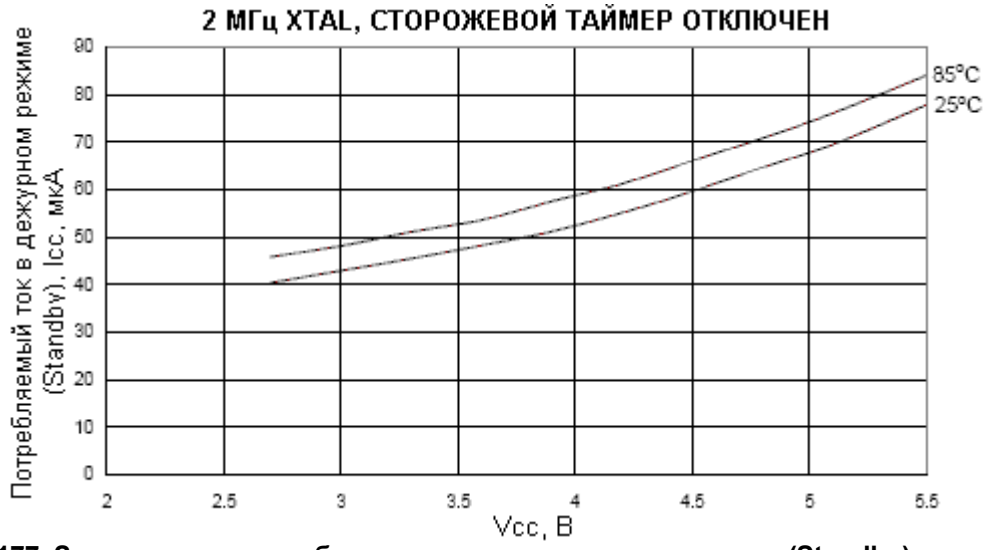


Рисунок 177. Зависимость потребляемого тока в дежурном режиме (Standby) от напряжения питания (тактирование XTAL 2МГц, сторожевой таймер отключен)

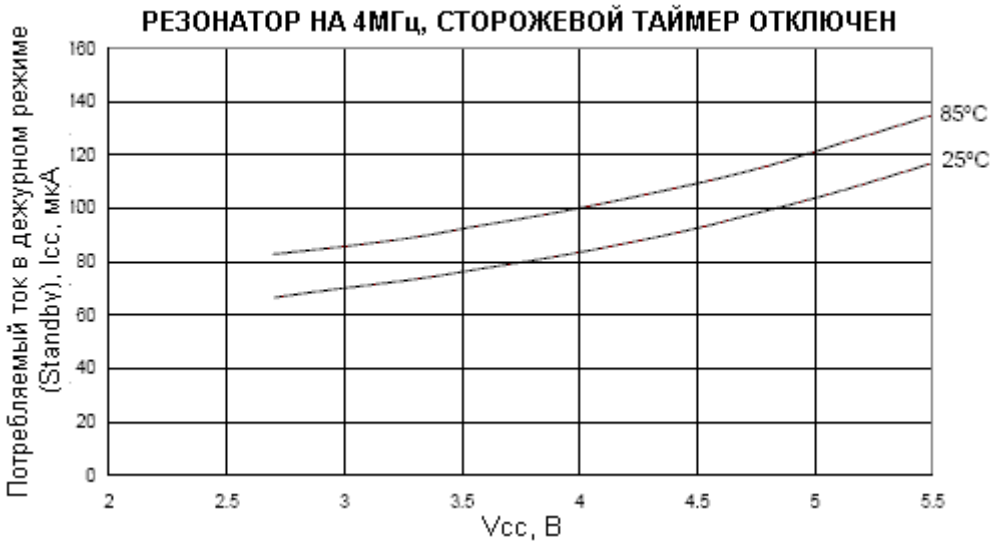


Рисунок 178. Зависимость потребляемого тока в дежурном режиме (Standby) от напряжения питания (тактирование резонатором частотой 4МГц, сторожевой таймер отключен)

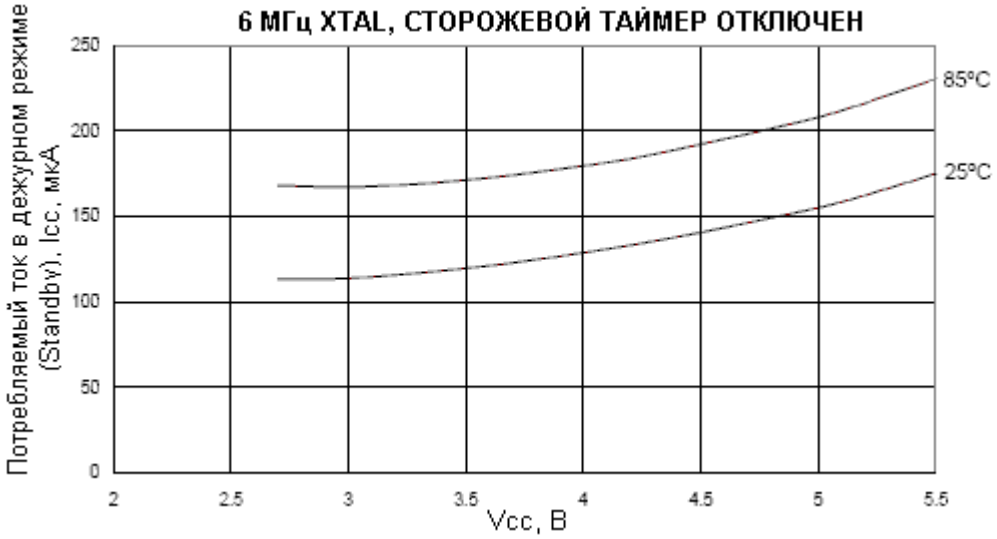


Рисунок 179. Зависимость потребляемого тока в дежурном режиме (Standby) от напряжения питания (тактирование XTAL 6МГц, сторожевой таймер отключен)

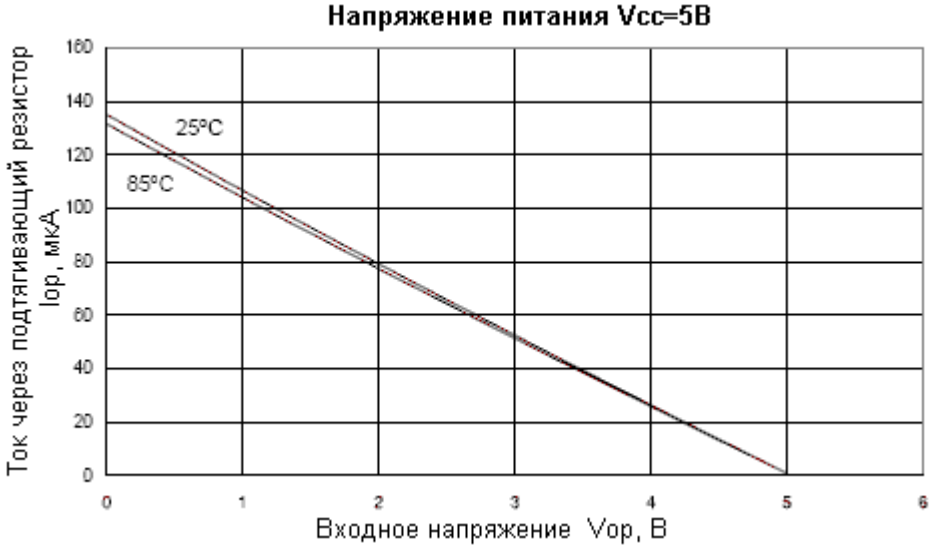


Рисунок 180. Зависимость тока через подтягивающий резистор портов ввода-вывода от входного напряжения при напряжении питания VCC = 5В

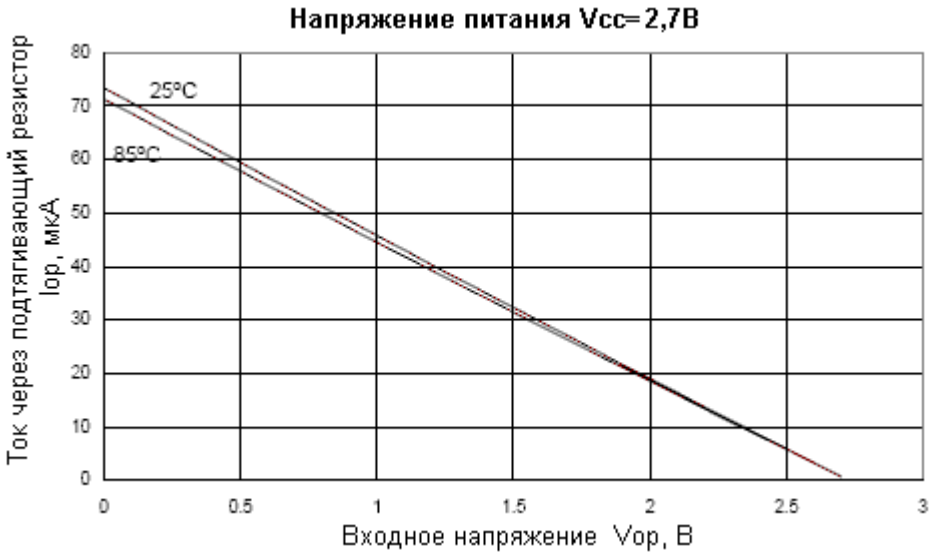


Рисунок 181. Зависимость тока через подтягивающий резистор портов ввода-вывода от входного напряжения при напряжении питания VCC = 2,7В

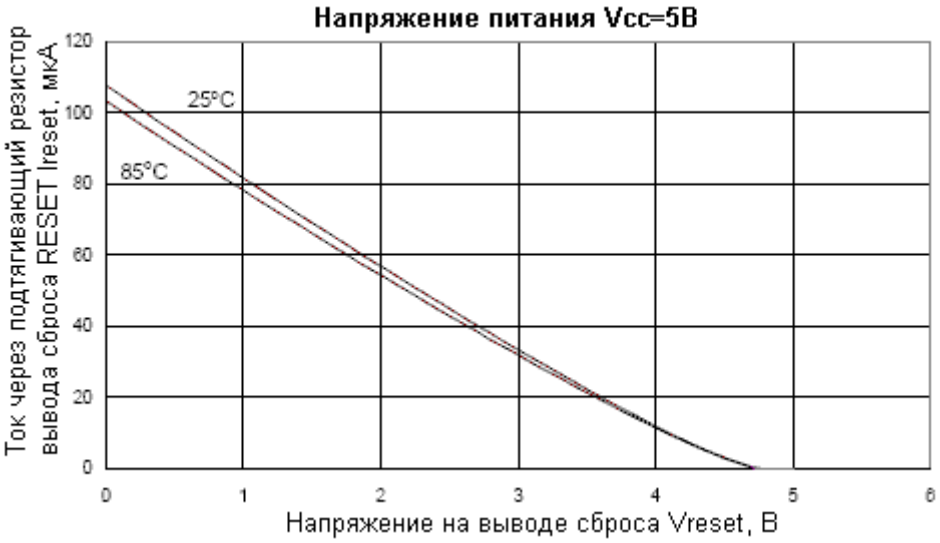


Рисунок 182. Зависимость тока через подтягивающий резистор вывода сброса (RESET) от входного напряжения при напряжении питания VCC = 5В

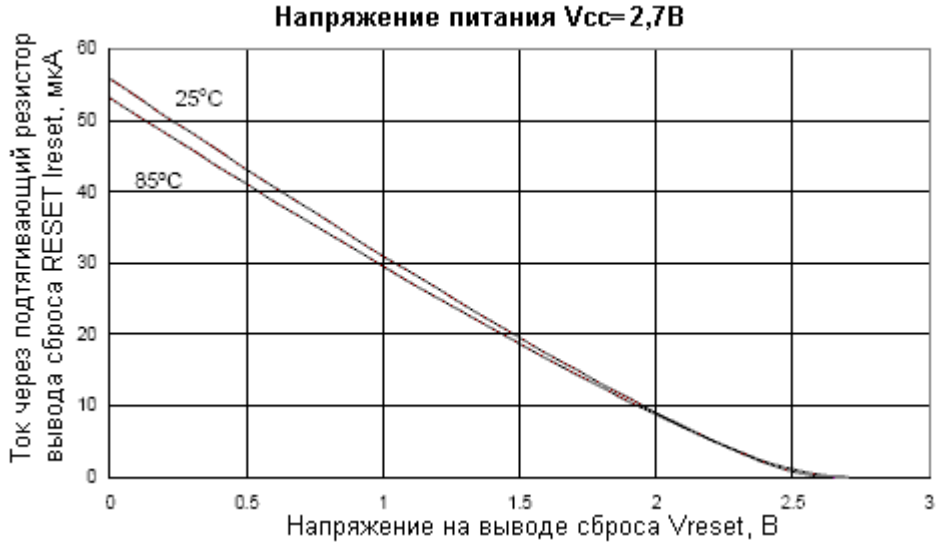


Рисунок 183. Зависимость тока через подтягивающий резистор вывода сброса (RESET) от входного напряжения при напряжении питания VCC = 2,7В

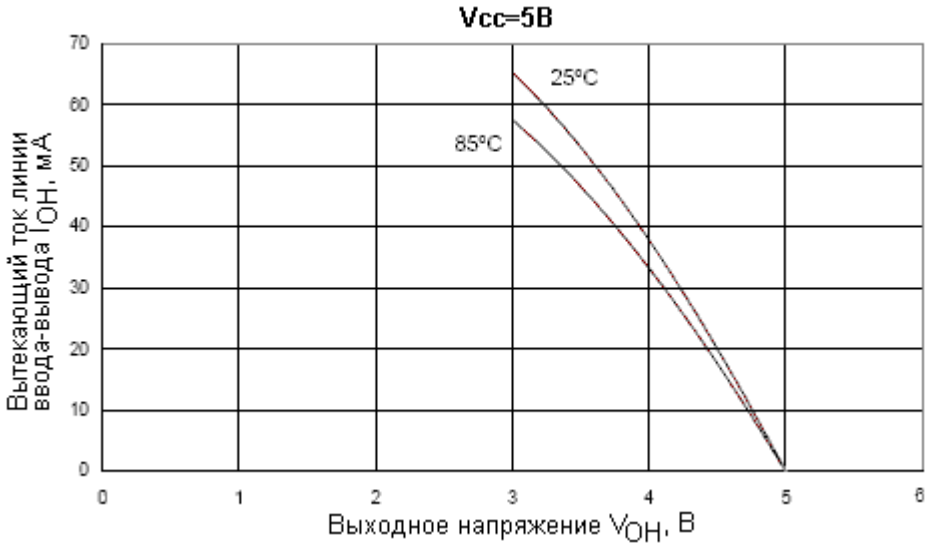


Рисунок 184. Зависимость вытекающего тока портов ввода-вывода от выходного напряжения при напряжении питания VCC = 5В

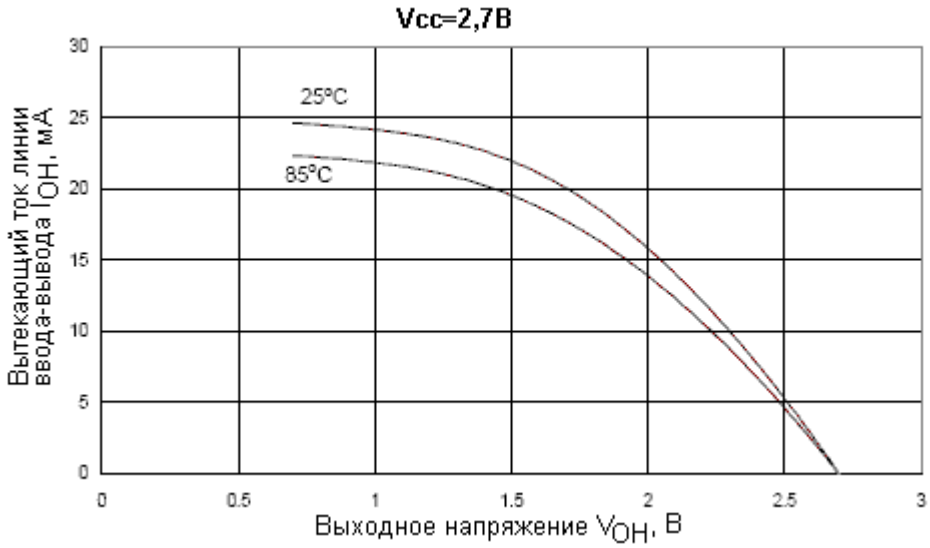


Рисунок 185. Зависимость вытекающего тока портов ввода-вывода от выходного напряжения при напряжении питания VCC = 2,7В

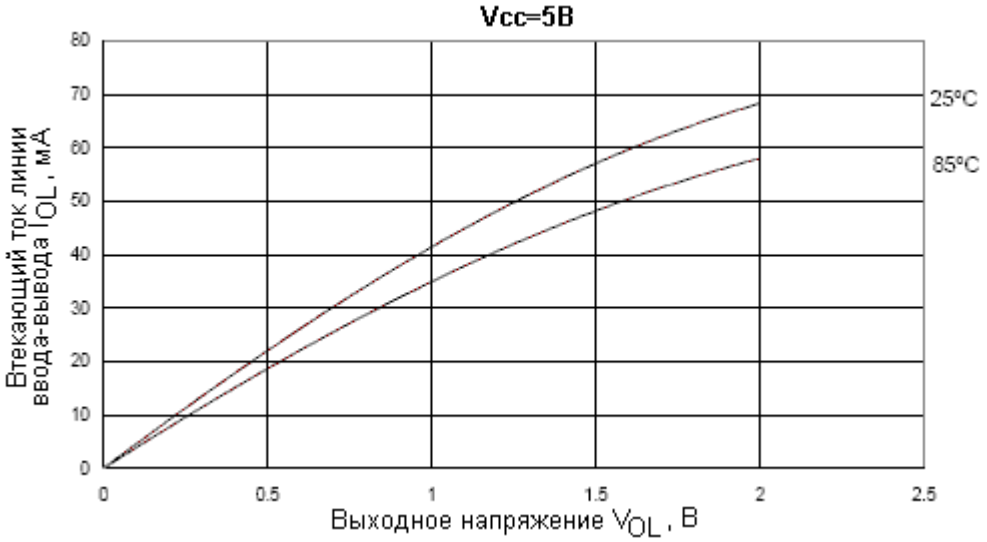


Рисунок 186. Зависимость втекающего тока портов ввода-вывода от выходного напряжения при напряжении питания VCC = 5В

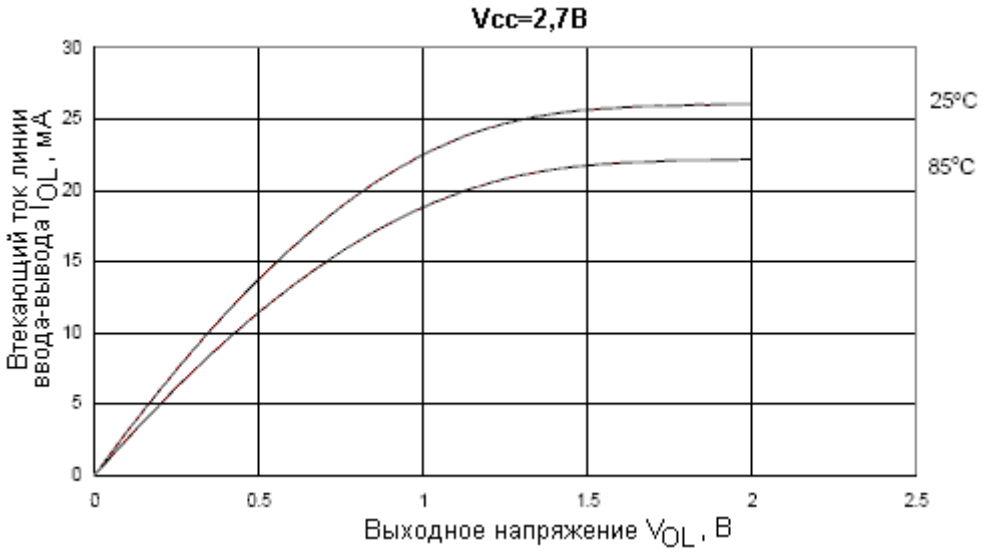


Рисунок 187. Зависимость втекающего тока портов ввода-вывода от выходного напряжения при напряжении питания VCC = 2,7В

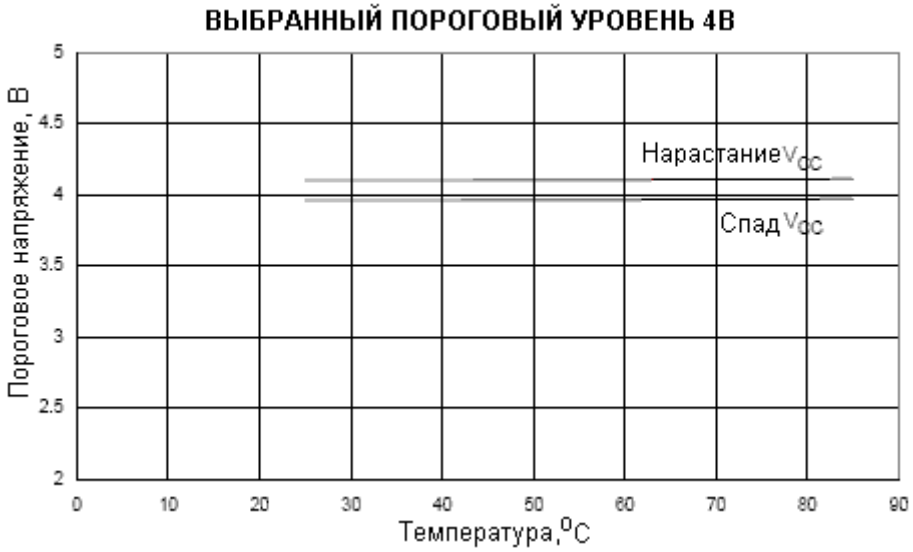


Рисунок 188. Зависимость порога срабатывания супервизора питания от температуры (выбранный порог срабатывания 4В)

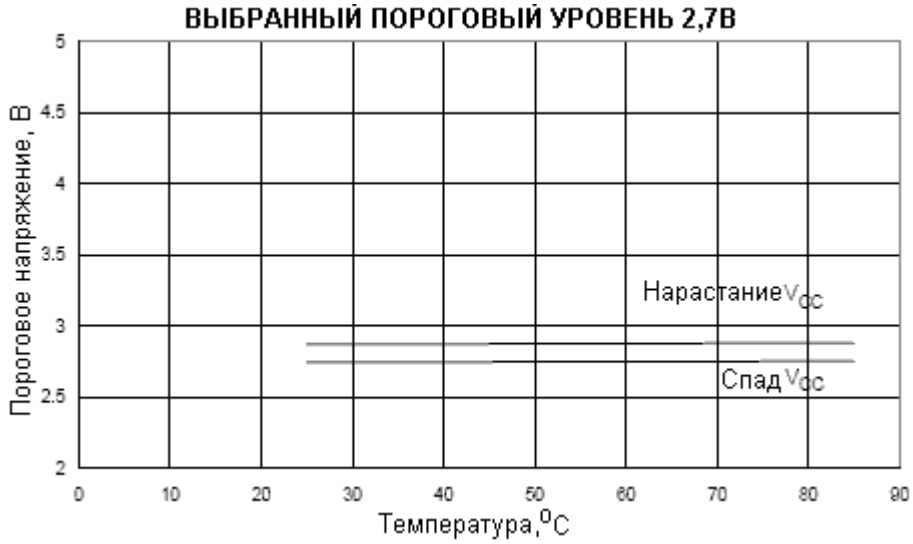


Рисунок 189. Зависимость порога срабатывания супервизора питания от температуры (выбранный порог срабатывания 2,7В)

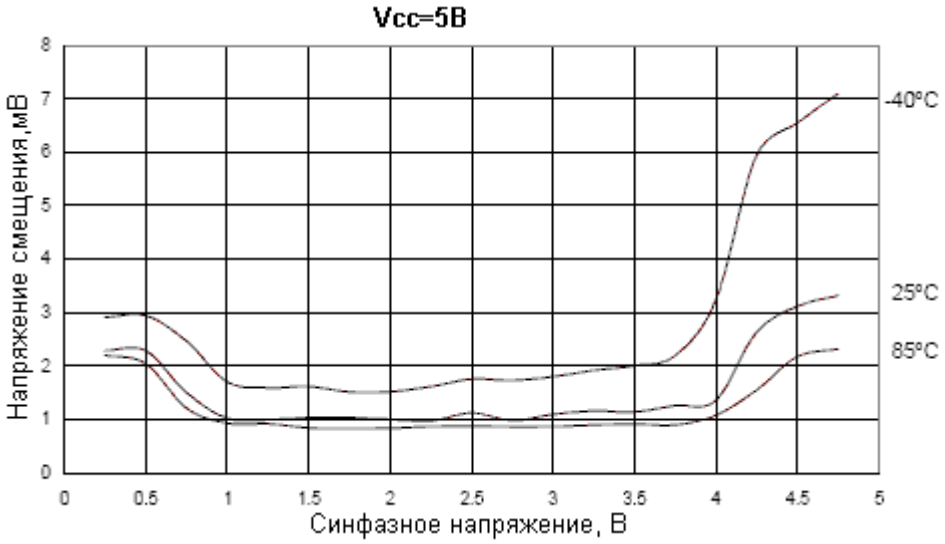


Рисунок 190. Зависимость напряжения смещения аналогового компаратора от синфазного напряжения при напряжения питания VCC = 5В

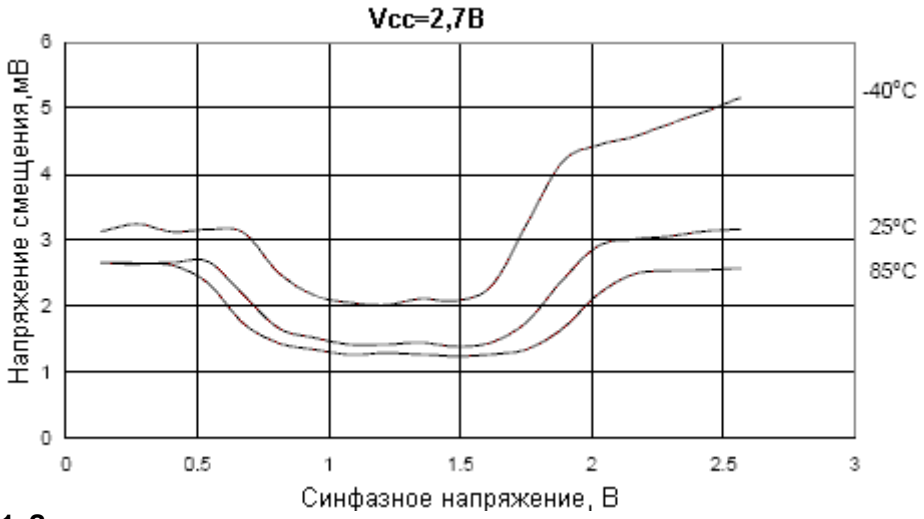


Рисунок 191. Зависимость напряжения смещения аналогового компаратора от синфазного напряжения при напряжения питания VCC = 2,7В

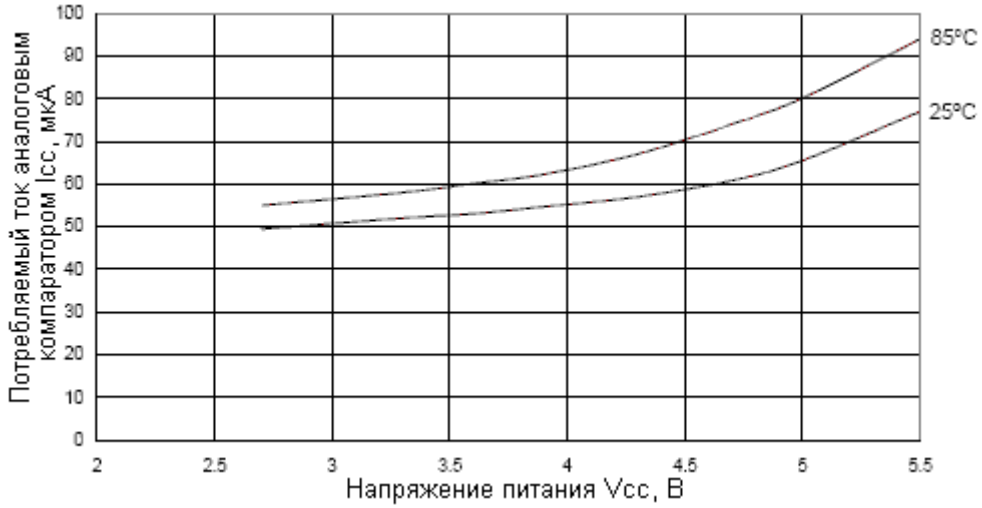


Рисунок 192. Зависимость потребляемого тока аналоговым компаратором от напряжения питания VCC

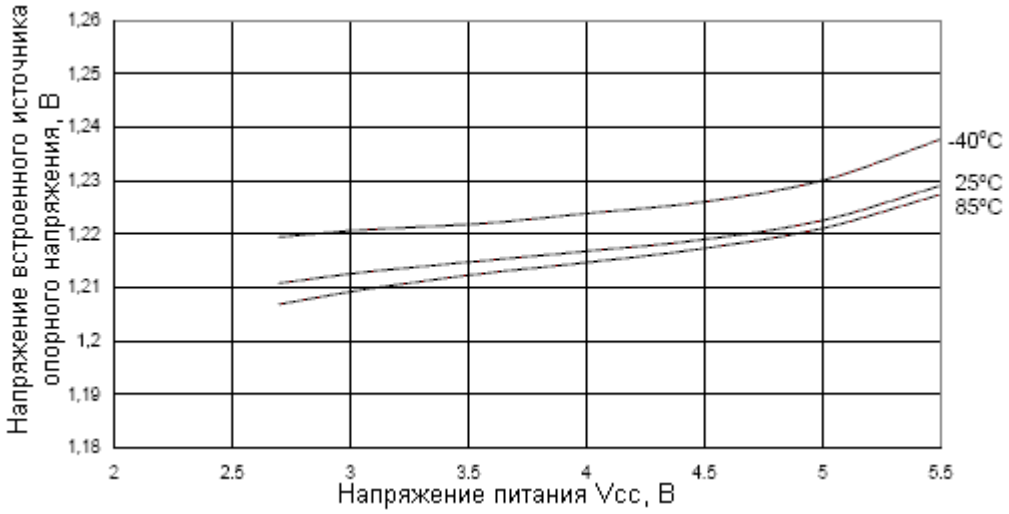


Рисунок 193. Зависимость напряжения встроенного источника опорного напряжения от напряжения питания VCC

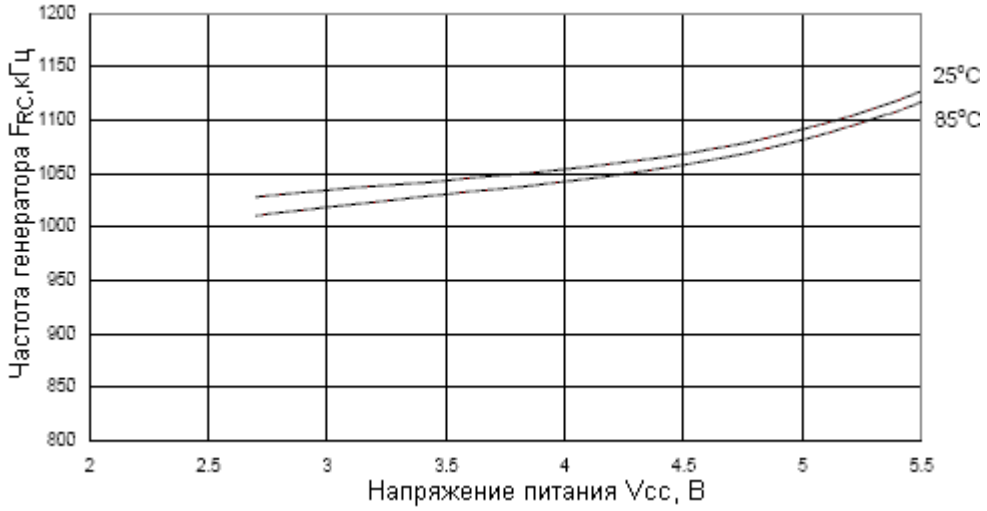


Рисунок 194. Зависимость частоты генератора сторожевого таймера от напряжения питания VCC

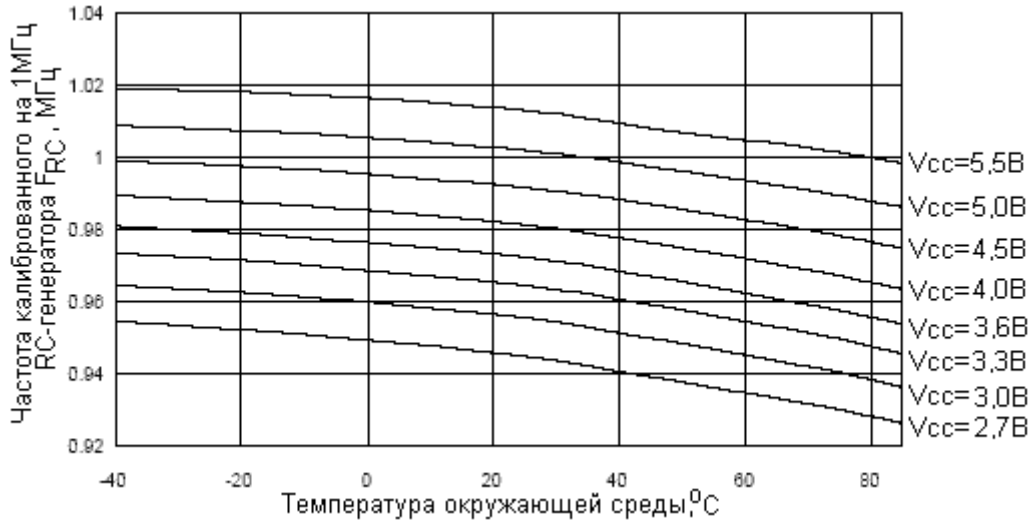


Рисунок 195. Зависимость частоты RC-генератора от температуры (микроконтроллеры откалиброваны на частоту 1МГц при напряжении питания Vcc = 5В и температуре T=25°C)

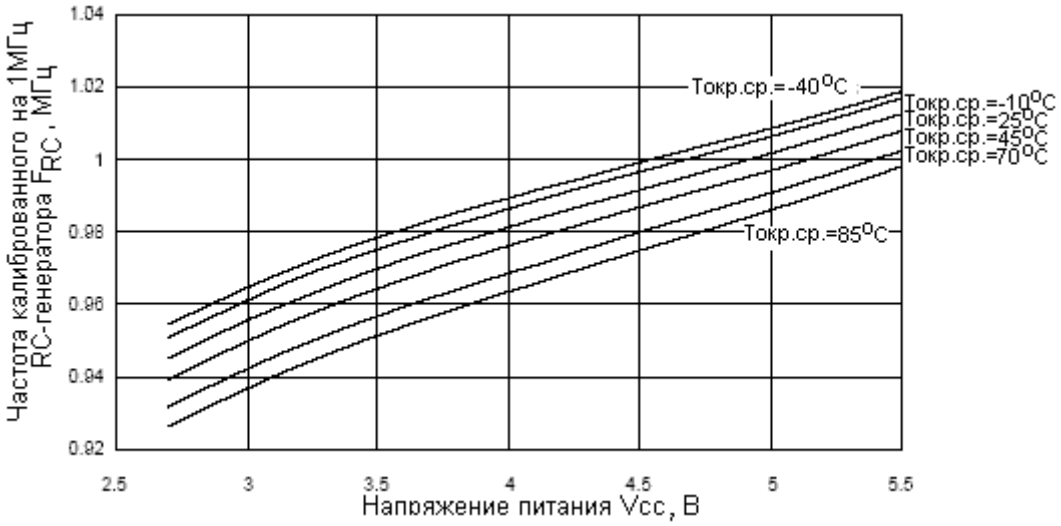


Рисунок 196. Зависимость частоты RC-генератора от напряжения питания (микроконтроллеры откалиброваны на частоту 1МГц при напряжении питания Vcc = 5В и температуре T=25°C)

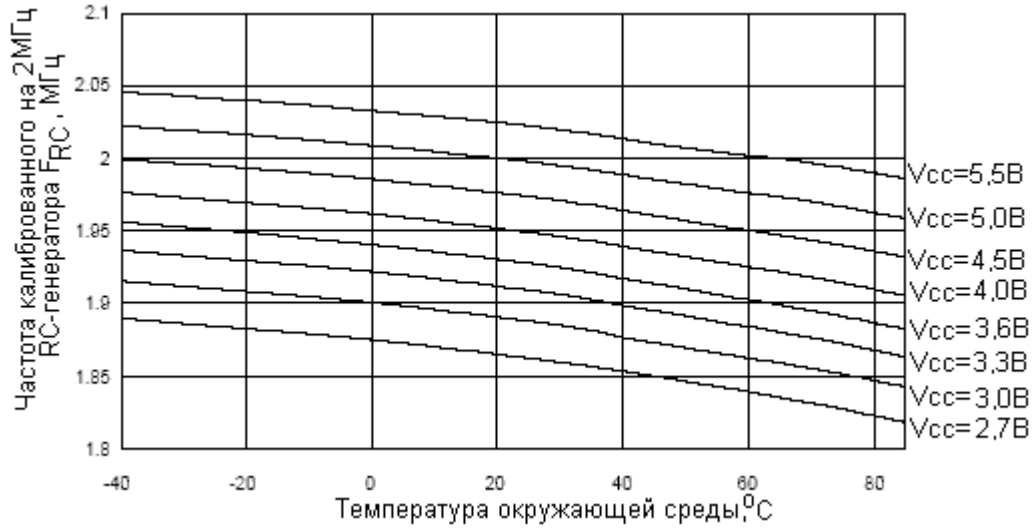


Рисунок 197. Зависимость частоты RC-генератора от температуры (микроконтроллеры откалиброваны на частоту 2МГц при напряжении питания Vcc = 5В и температуре T=25°C)

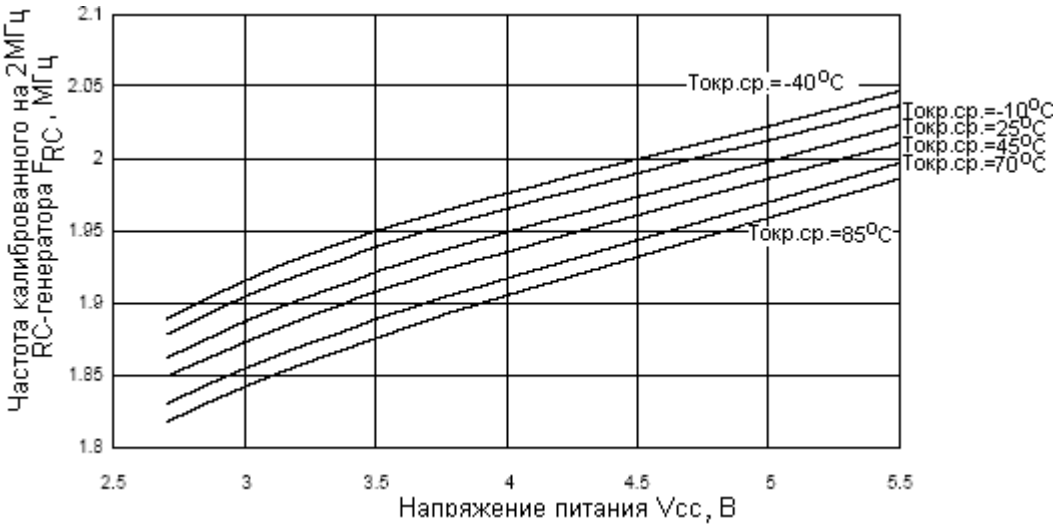


Рисунок 198. Зависимость частоты RC-генератора от напряжения питания (микроконтроллеры откалиброваны на частоту 2МГц при напряжении питания Vcc = 5В и температуре T=25°C)

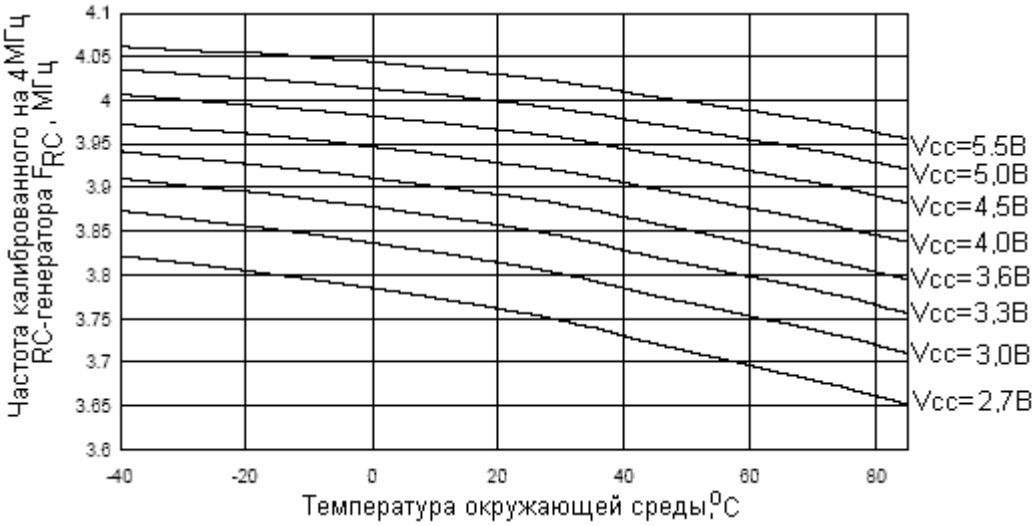


Рисунок 199. Зависимость частоты RC-генератора от температуры (микроконтроллеры откалиброваны на частоту 4МГц при напряжении питания Vcc = 5В и температуре T=25°C)

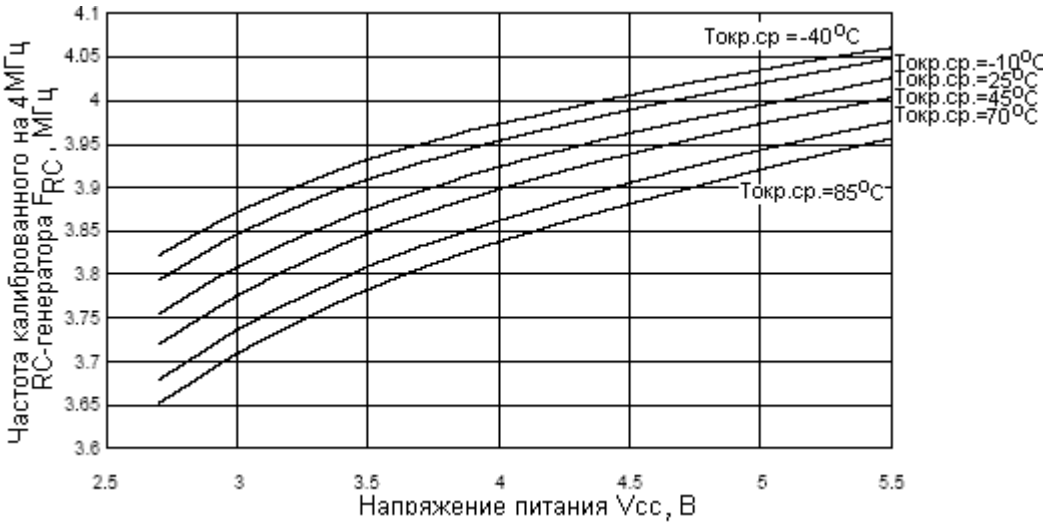


Рисунок 200. Зависимость частоты RC-генератора от напряжения питания (микроконтроллеры откалиброваны на частоту 4МГц при напряжении питания Vcc = 5В и температуре T=25°C)

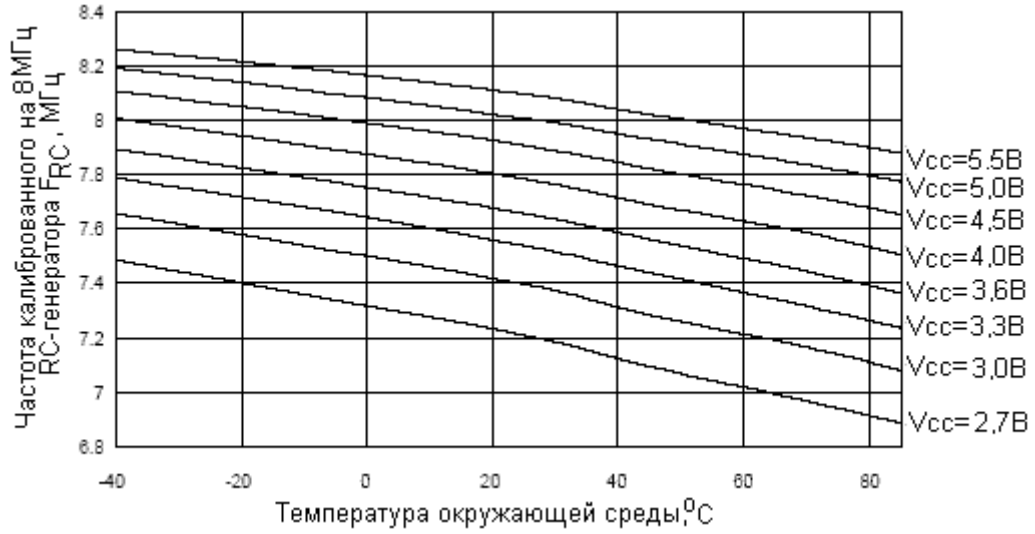


Рисунок 201. Зависимость частоты RC-генератора от температуры (микроконтроллеры откалиброваны на частоту 8МГц при напряжении питания Vcc = 5В и температуре T=25°C)

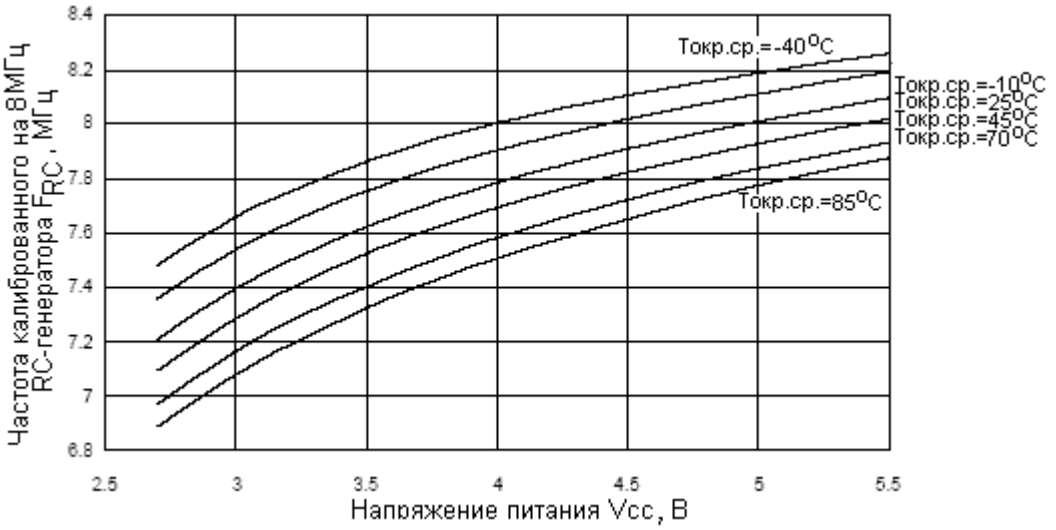


Рисунок 202. Зависимость частоты RC-генератора от напряжения питания (микроконтроллеры откалиброваны на частоту 8МГц при напряжении питания Vcc = 5В и температуре T=25°C)

Сводная таблица регистров

Адрес	Наименование	Разр.7	Разр.6	Разр.5	Разр.4	Разр.3	Разр.2	Разр.1	Разр.0
($FF)	Резерв	-	-	-	-	-	-	-	-
-	Резерв	-	-	-	-	-	-	-	-
($9E)	Резерв	-	-	-	-	-	-	-	-
($9D)	**UCSR1C**	-	UMSEL1	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1
($9C)	**UDR1**	Регистр данных УСАПП 1							
($9B)	**UCSR1A**	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1
($9A)	**UCSR1B**	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81
($99)	**UBRR1L**	Мл. байт регистра скорости связи УСАПП1							
($98)	**UBRR1H**	-	-	-	-	Ст. байт регистра скорости связи УСАПП1			
($97)	Резерв	-	-	-	-	-	-	-	-
($96)	Резерв	-	-	-	-	-	-	-	-
($95)	**UCSR0C**	-	UMSEL0	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0
($94)	Резерв	-	-	-	-	-	-	-	-
($93)	Резерв	-	-	-	-	-	-	-	-
($92)	Резерв	-	-	-	-	-	-	-	-
($91)	Резерв	-	-	-	-	-	-	-	-
($90)	**UBRR0H**	-	-	-	-	Ст. байт регистра скорости связи УСАПП0			
($8F)	Резерв	-	-	-	-	-	-	-	-
($8E)	Резерв	-	-	-	-	-	-	-	-
($8D)	Резерв	-	-	-	-	-	-	-	-
($8C)	**TCCR3C**	FOC3A	FOC3B	FOC3C	-	-	-	-	-
($8B)	**TCCR3A**	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30
($8A)	**TCCR3B**	ICNC3	ICES3	-	WGM33	WGM32	CS32	CS31	CS30
($89)	**TCNT3H**	Таймер-счетчик 3 - Старший байт регистра счетчика							
($88)	**TCNT3L**	Таймер-счетчик 3 - Младший байт регистра счетчика							
($87)	**OCR3AH**	Таймер-счетчик 3 - Ст. байт регистра A порога сравнения							
($86)	**OCR3AL**	Таймер-счетчик 3 - Мл. байт регистра A порога сравнения							
($85)	**OCR3BH**	Таймер-счетчик 3 - Ст. байт регистра B порога сравнения							
($84)	**OCR3BL**	Таймер-счетчик 3 - Мл. байт регистра B порога сравнения							
($83)	**OCR3CH**	Таймер-счетчик 3 - Ст. байт регистра C порога сравнения							
($82)	**OCR3CL**	Таймер-счетчик 3 - Мл. байт регистра C порога сравнения							
($81)	**ICR3H**	Таймер-счетчик 3 - Ст. байт регистра захвата							
($80)	**ICR3L**	Таймер-счетчик 3 - Мл. байт регистра захвата							
($7F)	Резерв	-	-	-	-	-	-	-	-
($7E)	Резерв	-	-	-	-	-	-	-	-
($7D)	**ETIMSK**	-	-	TICIE3	OCIE3A	OCIE3B	TOIE3	OCIE3C	OCIE1C
($7C)	**ETIFR**	-	-	ICF3	OCF3A	OCF3B	TOV3	OCF3C	OCF1C
($7B)	Резерв	-	-	-	-	-	-	-	-
($7A)	**TCCR1C**	FOC1A	FOC1B	FOC1C	-	-	-	-	-
($79)	**OCR1CH**	Таймер-счетчик 1 - Ст. байт регистра C порога сравнения							
($78)	**OCR1CL**	Таймер-счетчик 1 - Мл. байт регистра C порога сравнения							
($77)	Резерв	-	-	-	-	-	-	-	-
($76)	Резерв	-	-	-	-	-	-	-	-
($75)	Резерв	-	-	-	-	-	-	-	-
($74)	**TWCR**	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
($73)	**TWDR**	Регистр данных двухпроводного последовательного интерфейса							

($72)	**TWAR**	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
($71)	**TWSR**	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0
($70)	**TWBR**	Регистр задания скорости связи двухпров. послед. интерфейса							
($6F)	**OSCCAL**	Регистр калибровки генератора							
($6E)	Резерв	-	-	-	-	-	-	-	-
($6D)	**XMCRA**	-	SRL2	SRL1	SRL0	SRW01	SRW00	SRW11	-
($6C)	**XMCRB**	-	-	-	-	-	XMM2	XMM1	XMM0
($6B)	Резерв	-	-	-	-	-	-	-	-
($6A)	**EICRA**	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00
($69)	Резерв	-	-	-	-	-	-	-	-
($68)	**SPMCSR**	SPMIE	RWWSB	-	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN
($67)	Резерв	-	-	-	-	-	-	-	-
($66)	Резерв	-	-	-	-	-	-	-	-
($65)	**PORTG**	-	-	-	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0
($64)	**DDRG**	-	-	-	DDG4	DDG3	DDG2	DDG1	DDG0
($63)	**PING**	-	-	-	PING4	PING3	PING2	PING1	PING0
($62)	**PORTF**	PORTF7	PORTF6	PORTF5	PORTF4	PORTF3	PORTF2	PORTF1	PORTF0
($61)	**DDRF**	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0
($60)	Резерв	-	-	-	-	-	-	-	-
$3F($5F)	**SREG**	I	T	H	S	V	N	Z	C
$3E($5E)	**SPH**	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8
$3D($5D)	**SPL**	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
$3C($5C)	**XDIV**	XDIVEN	XDIV6	XDIV5	XDIV4	XDIV3	XDIV2	XDIV1	XDIV0

Примечания:

1. Для совместимости с последующими версиями микроконтроллеров рекомендуется в резервные разряды записывать лог. 0. В резервные ячейки памяти не рекомендуется выполнять запись.
2. 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.

Набор инструкций

Мнемокод	Операнды	Описание	Действие	Флаги	Кол. машинных циклов
Арифметические и логические инструкции					
ADD1	Rd, Rr	Сложить два регистра	Rd <— Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Сложить два регистра с переносом	Rd <— Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Сложить слово с константой	Rdh:Rdl <— Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Вычесть два регистра	Rd <— Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Вычесть константу из регистра	Rd <— Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Вычесть два регистра с учетом переноса	Rd <— Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Вычесть константу из регистра с учетом переноса	Rd <— Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Вычесть константу из слова	Rdh:Rdl <— Rdh:Rdl-K	Z,C,N,V,S	2
AND	Rd, Rr	Логическое И между регистрами	Rd <— Rd · Rr	Z,N,V	1

ANDI	Rd, K	Логическое И между регистром и константой	Rd <— Rd · K	Z,N,V	1
OR	Rd, Rr	Логическое ИЛИ между регистрами	Rd <— Rd v Rr	Z,N,V	1
ORI	Rd, K	Логическое ИЛИ между регистром и константой	Rd <— Rd v K	Z,N,V	1
EOR	Rd, Rr	Искл. ИЛИ между регистрами	Rd <— Rd ⊕ Rr	Z,N,V	1
COM	Rd	Дополнение до 0b11111111 ($FF), инверсия	Rd <— $FF - Rd	Z,C,N,V	1
NEG	Rd	Дополнение до 0b00000000 ($00)	Rd <— $00 - Rd	Z,C,N,V,H	1
SBR	Rd, K	Установка бит (бита) в регистре	Rd <— Rd v K	Z,N,V	1
CBR	Rd, K	Сброс бит (бита) в регистре	Rd <— Rd · ($FF - K)	Z,N,V	1
INC	Rd	Инкремент	Rd <— Rd + 1	Z,N,V	1
DEC	Rd	Декремент	Rd <— Rd - 1	Z,N,V	1
TST	Rd	Проверка на ноль или минус	Rd <— Rd . Rd	Z,N,V	1
CLR	Rd	Сброс регистра	Rd <— Rd ⊕ Rd	Z,N,V	1
SER	Rd	Установка регистра	Rd <— $FF	Нет	1
MUL	Rd, Rr	Умножение без знака	R1:R0 <— RdxRr	Z, C	2
MULS	Rd, Rr	Умножение со знаком	R1:R0 <— RdxRr	Z, C	2
MULSU	Rd, Rr	Умножение знакового с беззнаковым числом	R1:R0 <— Rd x Rr	Z, C	2
FMUL	Rd. Rr	Дробное умножение без знака	R1:R0 <— (RdxRr) << 1	Z, C	2
FMULS	Rd, Rr	Дробное умножение со знаком	R1:R0 <— (RdxRr) << 1	Z, C	2
FMULSU	Rd, Rr	Дробное умножение знакового с беззнаковым числом	R1:R0 <— (RdxRr) << 1	Z, C	2
Инструкции перехода					
RJMP	k	Относительный переход	PC <— PC + k +1	Нет	2
IJMP		Косвенный переход по указателю (Z)	PC <— Z	Нет	2
JMP	k	Безусловный переход	PC <— k	Нет	3
RCALL	k	Относительный вызов процедуры	PC <— PC + k + 1	Нет	3
ICALL		Косвенный вызов процедуры по указателю (Z)	PC <— Z	Нет	3
CALL	k	Безусловный вызов процедуры	PC <— k	Нет	4
RET		Возврат из подпрограммы	PC <— STACK	Нет	4
RETI		Возврат из прерывания	PC <— STACK	I	4
CPSE	Rd,Rr	Сравнение и пропуск, если равно if (Rd = Rr)	PC <— PC + 2 или 3	Нет	1/2/3
CP	Rd,Rr	Сравнение	Rd-Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Сравнение с учетом переноса	Rd - Rr-C	Z, N,V,C,H	1
CPI	Rd,K	Сравнение регистра с константой	Rd-K	Z, N,V,C,H	1
SBRC	Rr,b	Пропуск, если бит в регистре сброшен	if (Rr(b)=O)PC <— PC + 2 или 3	Нет	1 /2/3
SBRS	Rr, b	Пропуск, если бит в регистре установлен	if (Rr(b)=1)PC <— PC+ 2 или 3	Нет	1/2/3
SBIC	P, b	Пропуск, если бит в регистре	if (P(b)=O)PC <— PC	Нет	1 /2/3

		ввода-вывода сброшен	+ 2 или 3		
SBIS	P, b	Пропуск, если бит в регистре ввода-вывода установлен	if (P(b)=1)PC <— PC + 2 или 3	Нет	1 /2/3
BRBS	s, k	Переход, если флаг состояния установлен	if (SREG(s) = 1) then PC <— PC+k + 1	Нет	1/2
BRBC	s, k	Переход, если флаг состояния сброшен	if (SREG(s) = 0) then PC <— PC+k + 1	Нет	1 /2
BREQ	k	Переход, если равно	if (Z = 1) then PC <— PC + k + 1	Нет	1 /2
BRNE	k	Переход, если не равно	if (Z = 0) then PC <— PC + k + 1	Нет	1 /2
BRCS	k	Переход, если перенос установлен	if (C = 1)then PC <— PC + k+ 1	Нет	1 /2
BRCC	k	Переход, если перенос сброшен	if (C = 0) then PC <— PC + k + 1	Нет	1 /2
BRSH	k	Переход, если больше или равно	if (C = 0) then PC <— PC + k + 1	Нет	1 /2
BRLO	k	Переход, если меньше	if (C = 1) then PC <— PC + k+ 1	Нет	1 /2
BRMI	k	Переход, если минус	if (N = 1)then PC <— PC + k + 1	Нет	1 /2
BRPL	k	Переход, если плюс	if (N = 0) then PC <— PC + k + 1	Нет	1 /2
BRGE	k	Переход, если больше или равно с учетом знака	if (N e V= 0) then PC <— PC + k + 1	Нет	1 /2
BRLT	k	Переход, если меньше нуля с учетом знака	if (N e V= 1) then PC <— PC + k + 1	Нет	1 /2
BRHS	k	Переход, если флаг H установлен	if (H = 1)then PC <— PC + k + 1	Нет	1 /2
BRHC	k	Переход, если флаг H сброшен	if (H = 0) then PC <— PC + k + 1	Нет	1 /2
BRTS	k	Переход, если флаг T установлен	if (T = 1)then PC <— PC + k +1	Нет	1 /2
BRTC	k	Переход, если флаг T сброшен	if (T = 0) then PC <— PC + k + 1	Нет	1 /2
BRVS	k	Переход, если флаг V установлен	if (V = 1)then PC <— PC + k+ 1	Нет	1 /2
BRVC	k	Переход, если флаг V сброшен	if (V = 0) then PC <— PC + k + 1	Нет	1 /2
BRIE	k	Переход, если прерывания разрешены	if (I = 1)then PC <— PC + k + 1	Нет	1 /2
BRID	k	Переход, если прерывания запрещены	if (I = 0) then PC <— PC + k + 1	Нет	1 /2
Инструкции передачи данных					
MOV	Rd, Rr	Запись из регистра в регистр	Rd <— Rr	Нет	1
MOVW	Rd, Rr	Перезапись слова между регистрами	Rd+1:Rd <— Rr+1:Rr	Нет	1
LDI	Rd, K	Запись константы в регистр	Rd <— K	Нет	1
LD	Rd, X	Косвенное считывание из памяти в регистр	Rd <— (X)	Нет	2
LD	Rd, X+	Косвенное считывание из памяти в регистр и инкр.	Rd <— (X), X <— X + 1	Нет	2
LD	Rd,-X	Предварительный декремент, а затем косвенное считывание из памяти в регистр	X <— X - 1, Rd <— (X)	Нет	2

LD	Rd,Y	Косвенное считывание из памяти в регистр	Rd <— (Y)	Нет	2
LD	Rd, Y+	Косвенное считывание из памяти в регистр и инкр.	Rd <— (Y), Y <— Y + 1	Нет	2
LD	Rd,-Y	Предварительный декремент, а затем косвенное считывание из памяти в регистр	Y <— Y - 1, Rd <— (Y)	Нет	2
LDD	Rd,Y+q	Косвенное считывание из памяти в регистр со смещением	Rd <— (Y + q)	Нет	2
LD	Rd, Z	Косвенное считывание из памяти в регистр	Rd <— (Z)	Нет	2
LD	Rd, Z+	Косвенное считывание из памяти в регистр и инкр.	Rd <— (Z),Z <— Z+1	Нет	2
LD	Rd.-Z	Предварительный декремент, а затем косвенное считывание из памяти в регистр	Z <— Z - 1, Rd <— (Z)	Нет	2
LDD	Rd, Z+q	Косвенное считывание из памяти в регистр со смещением	Rd <— (Z + q)	Нет	2
LDS	Rd, k	Непосредственное чтение из ОЗУ в регистр	Rd <— (k)	Нет	2
ST	X, Rr	Косвенная запись	(X) <— Rr	Нет	2
ST	X+, Rr	Косвенная запись и послед. инкремент	(X) <— Rr, X <— X + 1	Нет	2
ST	-X, Rr	Предв. декремент и косвенная запись	X <— X - 1, (X) <— Rr	Нет	2
ST	Y, Rr	Косвенная запись	(Y) <— Rr	Нет	2
ST	Y+, Rr	Косвенная запись и послед. инкремент	(Y) <— Rr, Y <— Y + 1	Нет	2
ST	-Y, Rr	Предв. декремент и косвенная запись	Y <— Y - 1, (Y) <— Rr	Нет	2
STD	Y+q,Rr	Косвенная запись со смещением	(Y + q) <— Rr	Нет	2
ST	Z, Rr	Косвенная запись	(Z) <— Rr	Нет	2
ST	Z+, Rr	Косвенная запись и послед. инкремент	(Z) <— Rr, Z <— Z + 1	Нет	2
ST	-Z. Rr	Предв. декремент и косвенная запись	Z <— Z - 1, (Z) <— Rr	Нет	2
STD	Z+q,Rr	Косвенная запись со смещением	(Z + q) <— Rr	Нет	2
STS	k, Rr	Непосредственная запись в ОЗУ	(k) <— Rr	Нет	2
LPM		Чтение из памяти программ	R0 <— (Z)	Нет	3
LPM	Rd, Z	Чтение из памяти программ	Rd <— (Z)	Нет	3
LPM	Rd, Z+	Чтение из памяти программ и последующий инкремент	Rd <— (Z), Z <— Z+1	Нет	3
ELPM		Расширенное чтение из памяти программ	R0 <— (RAMPZ:Z)	Нет	3
ELPM	Rd, Z	Расширенное чтение из памяти программ	Rd <— (RAMPZ:Z)	Нет	3
ELPM	Rd, Z+	Расширенное чтение из памяти программ и последующие инкремент	Rd <— (RAMPZ:Z), RAMPZ:Z <— RAMPZ:Z+1	Нет	3
SPM		Запись в память программ	(Z) <— R1 :R0	Нет	-
IN	Rd, P	Считывание из порта ввода-вывода в регистр	Rd <— P	Нет	1

OUT	P, Rr	Запись из регистра в порт ввода-вывода	P <— Rd	Нет	1
PUSH	Rr	Помещение содержимого регистра в стек	STACK <— Rr	Нет	2
POP	Rd	Извлечение из стека в регистр	Rd <— STACK	Нет	2
Битовые инструкции и инструкции тестирования бит					
SBI	P,b	Установка бита в регистре ввода-вывода	I/O(P,b) <— 1	Нет	2
CBI	P,b	Сброс бита в регистре ввода-вывода	I/O(P,b) <— 0	Нет	2
LSL	Rd	Логический сдвиг влево	Rd(n+1) <— Rd(n), Rd(0) <— 0	Z,C,N,V	1
LSR	Rd	Логический сдвиг вправо	Rd(n) <— Rd(n+1), Rd(7) <— 0	Z,C,N,V	1
ROL	Rd	Вращение влево через перенос	Rd(0)<— C,Rd(n+1)<— Rd(n),C <— Rd(7)	Z,C,N,V	1
ROR	Rd	Вращение вправо через перенос	Rd(7) <— C,Rd(n)<— Rd(n+1),C <— Rd(0)	Z,C,N,V	1
ASR	Rd	Арифметический сдвиг вправо	Rd(n)<— Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Обмен тетрадами	Rd(3..0) <— Rd(7..4), Rd(7..4) <— Rd(3..0)	Нет	1
BSET	s	Установка флага регистра SREG	SREG(s) <— 1	SREG(s)	1
BCLR	s	Сброс флага регистра SREG	SREG(s) <— 0	SREG(s)	1
BST	Rr, b	Запись бита регистра в T	T <— Rr(b)	T	1
BLD	Rd, b	Чтение из T в бит регистра	Rd(b) <— T	Нет	1
SEC		Установка переноса	C <— 1	C	1
CLC		Сброс переноса	C <— 0	C	1
SEN		Установка флага N	N <— 1	N	1
CLN		Сброс флага N	N <— 0	N	1
SEZ		Установка флага нуля Z	Z <— 1	Z	1
CLZ		Сброс флага нуля Z	Z <— 0	Z	1
SEI		Общее разрешение прерываний	I<— 1	I	1
CLI		Общий запрет прерываний	I <— 0	I	1
SES		Установка флага S	S <— 1	S	1
CLS		Сброс флага S	S <— 0	S	1
SEV		Установка флага V в регистре SREG	V <— 1	V	1
CLV		Сброс флага V в регистре SREG	V <— 0	V	1
SET		Установка флага T в регистре SREG	T <— 1	T	1
CLT		Сброс флага T в регистре SREG	T <— 0	T	1
SEH		Установка флага H в регистре SREG	H <— 1	H	1
CLH		Сброс флага H в регистре SREG	H <— 0	H	1
Инструкции управления микроконтроллером					
NOP		Нет операции		Нет	1

SLEEP		Перевод в режим сна	(см. подробное описание режима сна)	Нет	1
WDR		Сброс сторожевого таймера	(см. подробное описание сторожевого таймера)	Нет	1
BREAK		Прерывание	Только для встроенной отладки	Нет	-

Информация для заказа

Макс. тактовая частота, МГц	Напряжение питания, В	Код заказа	Корпус(2)	Температурный диапазон
8	2.7 - 5.5В	ATmega128L-8AC ATmega128L-8MC	64A 64M1	Коммерческий (0°C…+70°C)
		ATmega128L-8AI ATmega128L-AU(2) ATmega128L-8MI ATmega128L-MU(2)	64A 64A 64M1 64M1	Промышленный (-40°C…+85°C)
16	4.5 - 5.5В	ATmega128-16AC ATmega128-16MC	64A 64M1	Коммерческий (0°C…+70°C)
		ATmega128-16AI ATmega128-6AU(2) ATmega128-16MI ATmega128-16MU(2)	64A 64A 64M1 64M1	Промышленный (-40°C…+85°C)

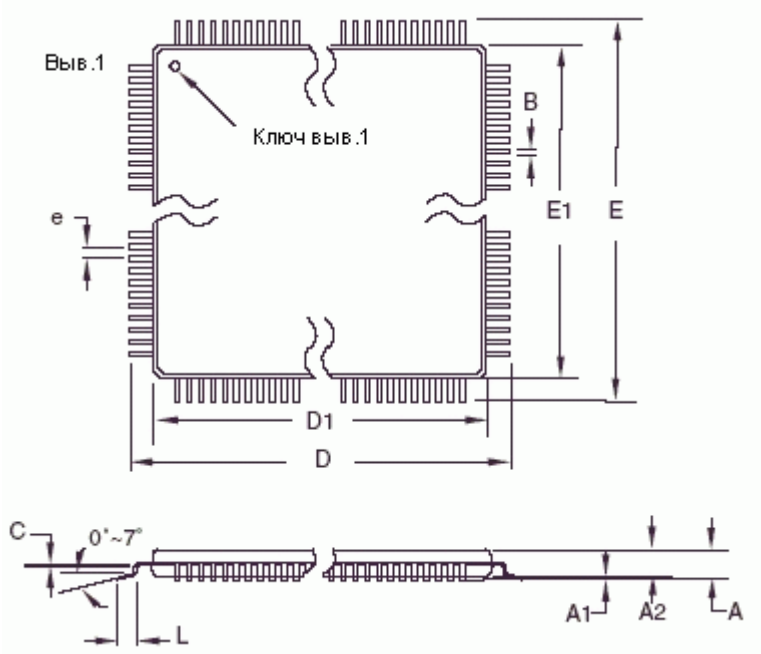
Прим.:

1. Микроконтроллер может также поставляться в виде кристалла. Для получения информации по заказу таких микроконтроллеров и минимальному количеству для заказа необходимо связаться с региональным торговым офисом компании Atmel.
2. Альтернативный корпус без содержания свинца, соответствует Европейской директиве по ограничению вредных веществ (директива RoHS). Данный корпус также не содержит галоидных соединений и полностью экологически безопасен.

Тип корпуса	
64A	64-выв., 14 x 14 x 1.0 мм пластиковый тонкопрофильный квадратный корпус TQFP
64M1	64-выв., 9 x 9 x 1.0 мм корпус MLF со сверх малым размером выводов

Информация по корпусам

64A



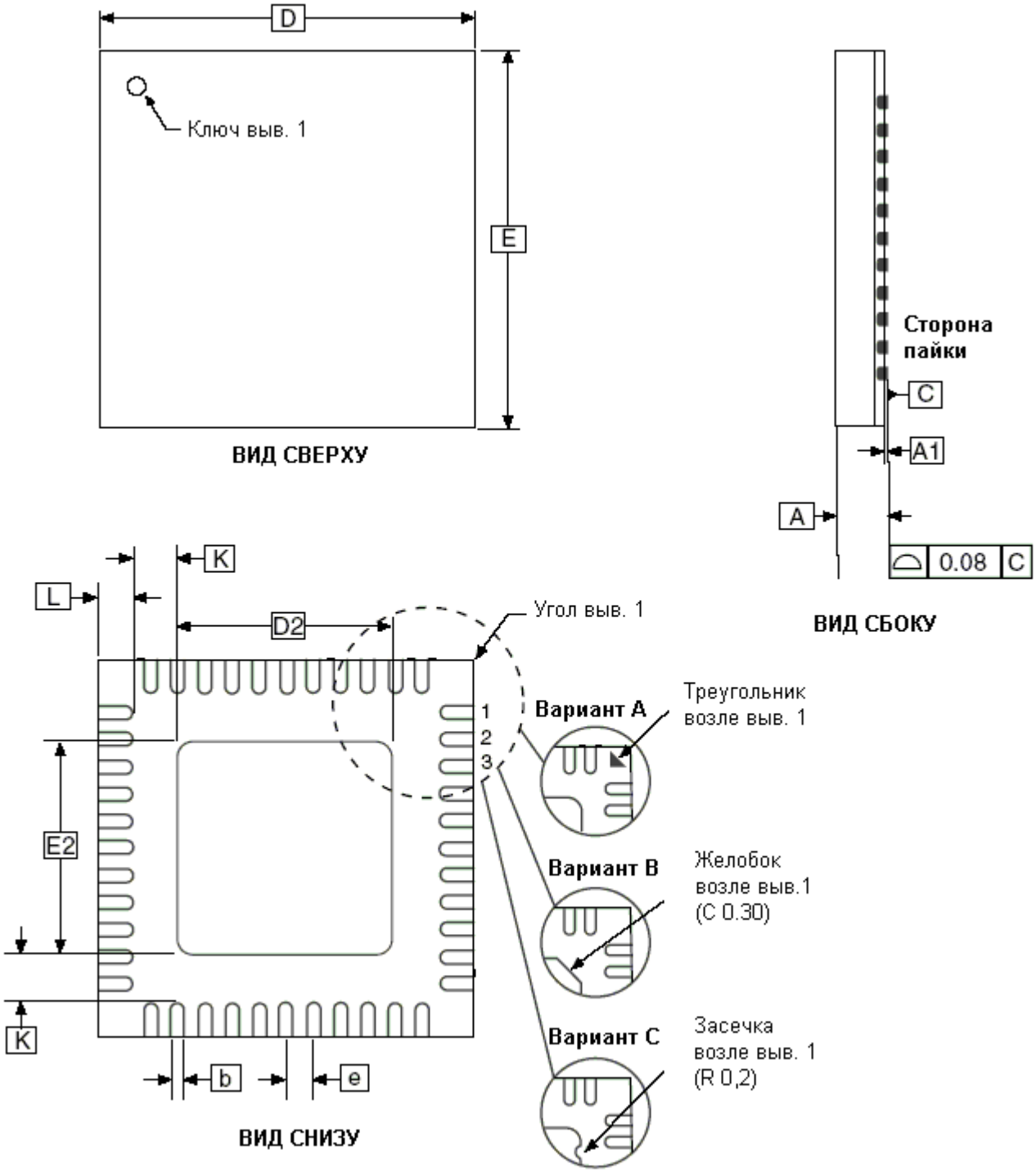
Размеры в мм

Обозн.	Мин.	Ном.	Макс.	Прим.
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	15.75	16.00	16.25	
D1	13.90	14.00	14.10	Прим.2
E	15.75	16.00	16.25	
E1	13.90	14.00	14.10	Прим.2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	Тип. 0.80			

Прим.:

1. Данный корпус соответствует стандарту JEDEC, ссылка MS-026, версия AEB.
2. Размеры D1 и E1 не учитывают выступы полимер-компаунда корпуса. Допустимый выступ - 0,25 мм на каждой стороне. Размеры D1 и E1 определяют габариты пластмассового корпуса с учетом выступом полимера.
3. Компланарность выводов не более 0,1 мм.

64M1


D
Ключ выв. 1
E
ВИД СВЕРХУ
Сторона пайки
C
A1
A
0.08
C
ВИД СБОКУ
K
L
D2
Угол выв. 1
1
2
3
E2
K
b
e
ВИД СНИЗУ
Вариант A
Треугольник возле выв. 1
Вариант B
Желобок возле выв.1 (C 0.30)
Вариант C
Засечка возле выв. 1 (R 0,2)

Размеры в мм

Обозн.	Мин.	Ном.	Макс.	Прим.
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
b	0.23	0.25	0.28	
D	Тип. 9.00			
D2	5.20	5.40	5.60	
E	Тип. 9.00			
E2	5.20	5.40	5.60	
e	Тип. 0.50			
L	0.35	0.40	0.45	
K	0.20	–	–	

Прим: Стандарт JEDEC MO-220, рис. 1, VMMD.