# Особенности:

- Используется AVR ® RISC-Архитектура
- AVR - Быстродействующая и малопотребляющая RISC Архитектура
- 120 Мощных Команд - большинство выполняется за 1 так ЦПУ
- 32 x 8 регистра общего назначения (32 регистра 8-ми разрядных)
- Полностью Статическая Операция
- Производительность до 20 МИЛЛИОНОВ КОМАНД В СЕКУНДУ на 20 МГЦ ЦПУ
- Энергонезависимая память данных и программ
- 2 КБ внутрисистемной энергонезависимой ФЛЭШ-памяти программ
- Выносливость ФЛЭШ-памяти 10 000 циклов записи/стирания
- 128-байтовая встроенная программируемая EEPROM память
- Выносливость EEPROM: 100 000 циклов записи/стирания
- 128-байтовая внутренняя SRAMпамять
- Программирование защитной блокировки для ФЛЭШ-прграмм и EEPROM-данных
- Периферийные Особенности:
- Один 8-битный Таймер/Счетчик с Отдельным Предделителем частоты и Режимом сравнения
- Один 16-битный Таймер/Счетчик с Отдельным Предделителем частоты и Режимом сравнения
- Четыре ШИМ Канала
- Встроенный в чип Аналоговый Компаратор
- Программируемый Сторожевой Таймер со встроенным Генератором
- USI - Универсальный Последовательный Интерфейс
- Полно-Дуплексный интерфейс - USART
- Дополнительные Особенности Микроконтроллера:
- debugWIRE-шина для Отладки На-чипе
- Внутрисистемное Программирование через SPI Порт
- Внешние и Внутренние Источники Прерывания
- Холостой режим (Low-power Idle отключает только ЦПУ), экономичный режим (Power-down отключает только генератор), и Режим Сна (Standby Modes оставляет включенным только Генератор)
- Усовершенствованная схема сброса при включении питания
- Программируемая Схема Защиты от пониженного напряжения питания Brown-out Detector (BOD)
- Внутренний Калиброванный Генератор
- Порты ввода/ вывода и Корпуса:
- 18 Программируемых линий ввода - вывода
- PDIP с 20 ножками, SOIC с 20 ножками, и MLF с 32 ножками
- Напряжения питания:
- 1.8 - 5.5V (ATtiny2313V)
- 2.7 - 5.5V (ATtiny2313)
- Таблица Производительности:
- ATtiny2313V: 0 - 4 МГЦ 1.8 - 5.5V, 0 - 10 МГЦ 2.7 - 5.5V
- ATtiny2313: 0 - 10 МГЦ 2.7 - 5.5V, 0 - 20 МГЦ 4.5 - 5.5V
- Типичное Потребление Энергии:
- Активный Режим:
  1 МГЦ, 1.8V: 230 µA
  32 kHz, 1.8V: 20 µA (при включенном генераторе)
  Экономичный режим:
  <0.1 µA в 1.8V

**8-bit AVR® Microcontroller with 2K Bytes In-System Programmable Flash**

**ATtiny2313/V**

**Preliminary**



| | | | | | | |
|---|---|---|---|---|---|---|
| | | г Шахты | iprito@mail.ru | | **ATtiny2313 документация (полная)** | Лист |
| | | ЮРГУЭС | Конев ДН | осень | | 1 |
| | | Магистратура | Подпись | 2006г | | |

# Конфигурация Выводов

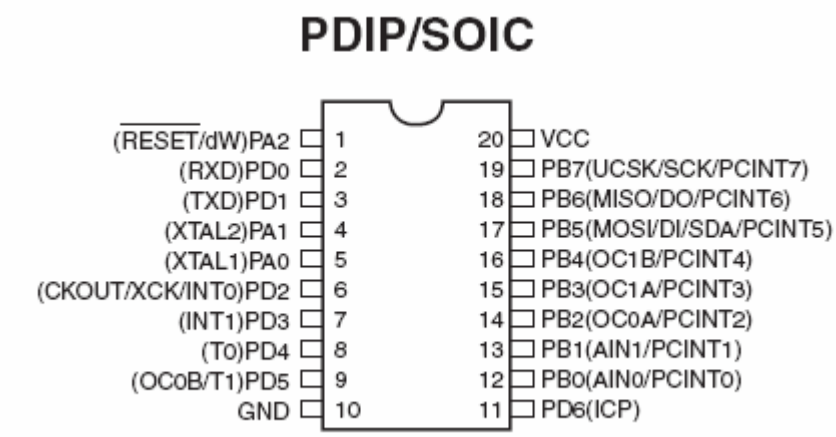



Рисунок 1 – Цоколевка ATtiny2313 в двух типах корпуса

# Краткий обзор

ATtiny2313 – малопотребляющий CMOS -8-битный микроконтроллер, основанный на AVR-усовершенствованной RISC-архитектуре. Выполняя мощные команды за один такт ЦПУ, ATtiny2313 достигает производительности, приближающейся к 1 МИЛЛИОНУ КОМАНД В СЕКУНДУ на 1МГЦ и позволяет системному проектировщику оптимизировать потребление мощности в компромиссе со скоростью обработки.

Прибор изготовлен, используя технологию Atmel долговременной памяти с высокой плотностью. Встроенная в чип внутрисистемная программируемая Флэш-память позволяет перепрограммировать память программ через SPI-(serial peripheral interface)последовательный интерфейс или через обычный программатор постоянной памяти. Объединяя 8-битный RISC ЦПУ со встроенной Самопрограммируемой Флэш-памятью на одной полупроводниковой Интегральной Схеме контроллер ATtiny2313 является мощным устройством обеспечивающим высокую гибкость
и дешевое решение для многих внедряемых приложений.

ATtiny2313 поддерживается программными инструментальными средствами разработки, включая: Си-компиляторы, Макроассемблеры, Программные Отладчик/Симуляторы, Внутрисхемные Эмуляторы и Оценочные Комплекты.

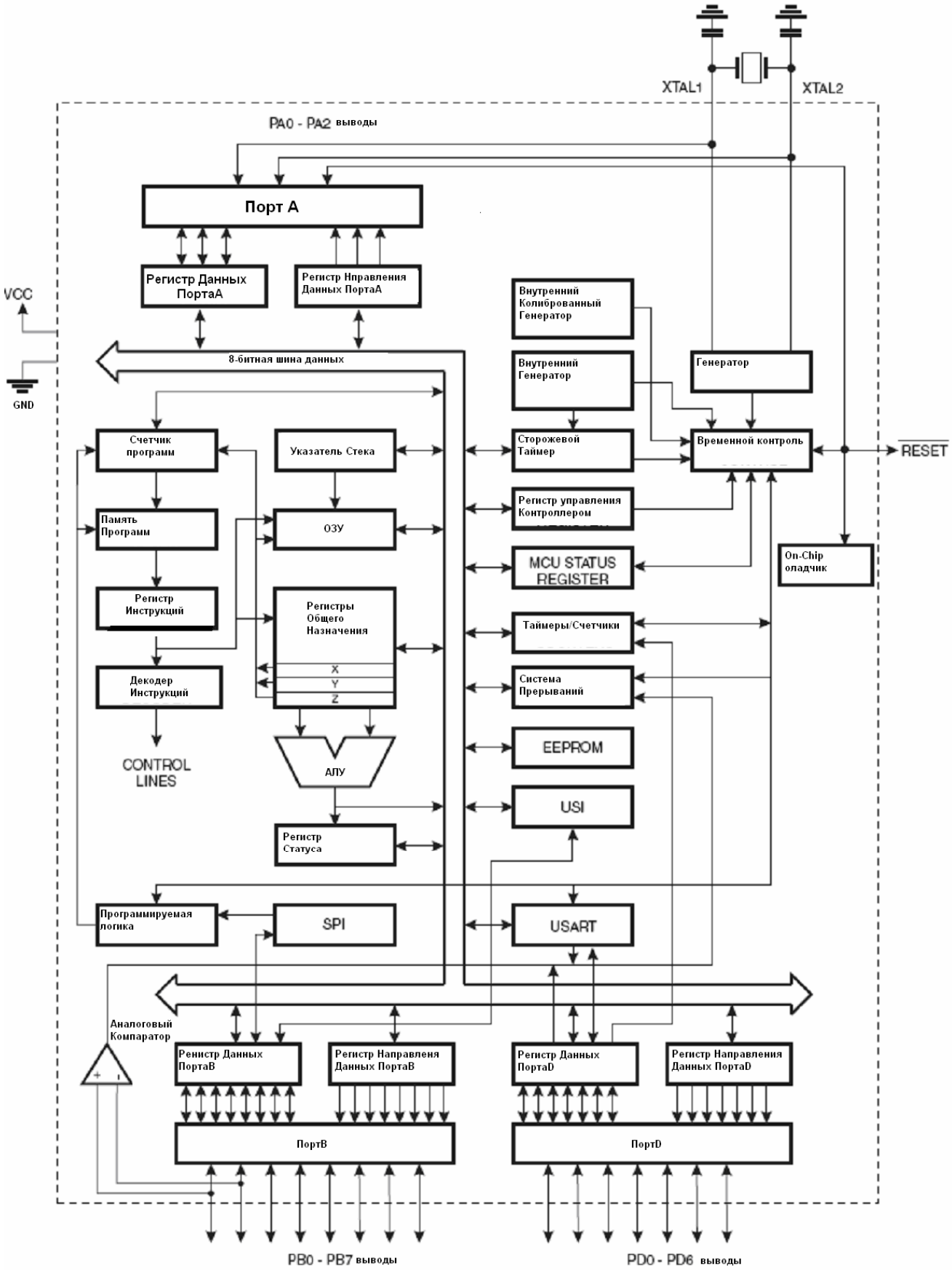



Рисунок 2 – Блок-Диаграма ATtiny2313 (внутренняя архитектура)

AVR ядро комбинирует богатый набор инструкций с 32 регистрами общего назначения. Все 32 регистра непосредственно связаны с Арифметико-Логическим Устройством (ALU), что позволяет двум независимым регистрам обращаться к одной инструкции и выполнять ее за 1цикл ЦПУ. RISC архитектура более эффективна по коду достигая прироста производительности в десять раз по сравнению с обычными CISC-микроконтроллерами.

ATtiny2313 имеет следующие особенности: 2кБ встроенной программируемой ФЛЭШ-памяти программ, 128 байт EEPROM, 128 байт SRAM, 18 общих линий ввода - вывода общего назначения, 32 регистра общего назначения, однопроводной Интерфейс для Отладки На-чипе, два настраиваемых Таймер/Счетчика с режимами сравнения, внутренние и внешние прерывания, программируемый последовательный интерфейс -USART, Универсальный Последовательный Интерфейс USI с Датчиком Начального Состояния, программируемый Сторожевой таймер с внутренним Генератором и три программно выбираемых режима низкого потребления энергии.

Холостой Режим (Idle mode) останавливает ЦПУ, но позволяет памяти SRAM, Таймерам/Счетчикам и Системе прерывания продолжать функционировать.

Режим Энергосбережения (Power Down mode) сохраняет содержание регистров, но останавливает Генератор, запрещает все другие функции контроллера до следующего прерывания или аппаратного сброса.

Режим сна (Standby mode) оставляет включенным только Генератор, пока остальные функции контроллера выключены, что позволяет осуществлять экономию энергии и одновременно с этим быстро запускать контроллер в работу.

# Описание Выводов

| | |
|---|---|
| **VCC** | «Цифровое» напряжение питания |
| **GND** | «Земля» |
| **Port A (PA2..PA0)** | Порт A это 3-битный двунаправленный порт ввода/вывода с внутренними подтягивающими(нагрузочными) резисторами (выбираемыми для каждого вывода). Выходные буферы Порта A имеют одинаковые характеристики с высокой нагрузочной способностью. Если выводы Порта A замыкаются на землю и включены нагрузочные резисторы то выводы являются источниками тока. Сразу после сброса контроллера выводы Порта A находятся в ВысокоИмпедансном состоянии (третьем состоянии) даже, если генератор не запущен. Порт A также обслуживает функции различных специальных елементов контроллера ATtiny2313 представленных на стр52. |
| **Port B (PB7..PB0)** | Порт A это 8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими(нагрузочными) резисторами (выбираемыми для каждого вывода). Выходные буферы Порта B имеют одинаковые характеристики с высокой нагрузочной способностью. Если выводы Порта B замыкаются на землю и включены нагрузочные резисторы то выводы являются источниками тока. Сразу после сброса контроллера выводы Порта B находятся в ВысокоИмпедансном состоянии (третьем состоянии) даже, если генератор не запущен. Порт B также обслуживает функции различных специальных элементов контроллера ATtiny2313 представленных на стр52. |
| **Port D (PD6..PD0)** | Порт A это 7-битный двунаправленный порт ввода/вывода с внутренними подтягивающими(нагрузочными) резисторами (выбираемыми для каждого вывода). Выходные буферы Порта D имеют одинаковые характеристики с высокой нагрузочной способностью. Если выводы Порта D замыкаются на землю и включены нагрузочные резисторы то выводы являются источниками тока. Сразу после сброса контроллера выводы Порта D находятся в ВысокоИмпедансном состоянии (третьем состоянии) даже, если генератор не запущен. Порт D также обслуживает функции различных специальных элементов контроллера ATtiny2313 представленных на стр52. |
| **RESET** | Вывод сброса контроллера. Низкий уровень на этом выводе длящийся дольше чем минимальная длительность импульса генерирует сброс, даже если ЦПУ не запущен. Минимальная длительность импульса приводится в Таблице 15 на странице 33. Более короткие импульсы не генерируют сброс в большинстве случаев. Ввод Сброса имеет дополнительные, т.е. альтернативные функции: PA2 и dW. |
| **XTAL1** | Этот Вывод является входом Инвертирующего Усилителя и входом для внутренней схемы вырвбатывающей тактовый сигал. XTAL1 имеет альтернативную функцию - PA0. |
| **XTAL2** | Этот вывод является выходом Инвертирующего Усилителя Генератора. XTAL2 имеет альтернативную функцию - PA1. |

# О примерах программного кода

Эта документация содержит простые примеры кода, которые кратко показывают , как использовать разные части контроллера. Эти примеры предполагают, что специальные заголовочные файлы будут включены перед компиляцией проекта. Имейте в виду, что не все производители/распространители C-компиляторов включают в заголовочные файлы описания битов, а обработчики прерываний могут быть по-разному реализованы в разных C-компиляторах.За более подробной информацией по этому поводу обращайтесь к документации на компилятор.

# Примечание

Типичные значения содержащиеся в этой длкументации базированы на моделировании и характеристиках других контроллерах этого семейства, произведенных на подобном технологическом процессе. Минимальные и Максимальные приведенные значения можно применять после прочтения всей документации.

# Ядро ЦПУ AVR

## Введение

Здесь в общем виде обсуждается архитектура AVR-ядра. Главная функция ядра ЦПУ -гарантировать правильное выполнение программ. По этой причине ЦПУ должен быть в состоянии обращаться к памяти, выполнять вычисления, контролировать периферию и обрабатывать прерывания.

## Обзор Архитектуры

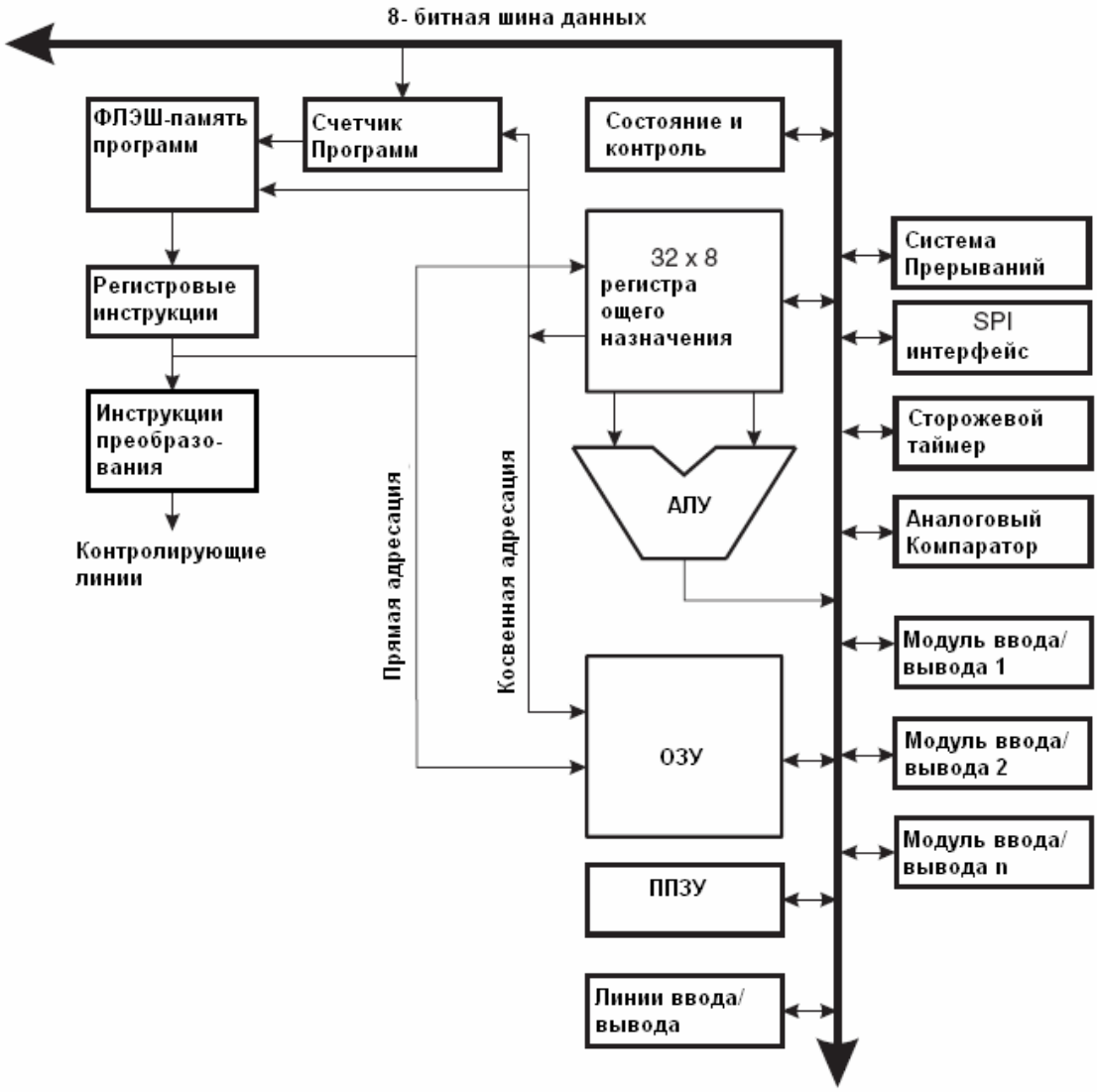



Рисунок 3 – Блок-Диаграмма AVR-архитектуры

Чтобы максимально улучшить характеристики, в AVR используется гарвардская архитектура - с раздельной памятью и шинами программ и данных. Инструкции в памяти программ выполнены с одноуровневой конвейерной обработкой данных. Пока одна инструкция выполняется следующая инструкция уже предвыбрана из памяти программ. Такая концепция позволяет инструкциям выполняться за каждый такт генератора. Память программ представляет собой встроенную ФЛЭШ-память с возвможносью перепрограммирования.

Регистровый файл быстрого доступа содержит 32 8-битных регистра общего назначения с временем доступа за один такт. Это позволяет устройстсву АЛУ выполнять операции за один такт системной частоты. В типичной операции арифметико-логического устройства, два операнда выводятся от Регистрового файла,

операция выполняется, а результат сохраняется обратно в Регистровый Файл – и все это за один такт.

Шесть из 32 регистров могут использоваться как три 16-разрядных косвенных регистра указателя адреса для Пространства Данных - позволяющих эффективную адресацию. Один из иэтих указателей адреса может быть использован как указатель адреса для просмотра таблиц во ФЛЭШ-памяти программ. Эта функция добавлена 16-битным регистрам X,Y и Z, описанным дальше.

АЛУ поддерживает арифметические и логические операции между регистрами и между константами и регистрами. Операция с одним регистром так же может быть выполнена в АЛУ. После арифметической операции Регистр Статуса обновляется для отображения информации о результате вычисления.

Процесс выполнения программы обеспечивает Условный и Безусловный переход а также вызов Инструкций способных прямо адресоваться по всему адресному пространству. Большинство AVR-инструкций имеют 16-битный формат Слова. Каждый адрес памяти программ содержит 16- или 32-битную инструкцию.

Во время выполнения прерываний и вызовов подпрограмм адрес возврата Счетчика Программ сохраняется в Стеке. Стек фактически расположен в пространстве данных ОЗУ и следовательно ограничен только размерами местной ОЗУ. Все программы Пользователя должны инициализировать(калибровать/определять/устанавливать) SP-Указатель Стека в подпрограмме обработки Сброса контроллера до! вызова подпрограмм и выполнения прерываний. Указатель Стека доступен для чтения/записи в области ввода/вывода. Данные ОЗУ могут быть легко доступны через Пять различных режимов адресации поддерживаемых в AVR архитектуре.

Пространство памяти в AVR архитектуре распределено линейно и поделено на различные области использования.

Гибконастраиваемый Модуль Прерываний имеет регистр управления расположенный в области памяти и используется в комбинации с Флагом Глобального Прерывания, расположенном в Регистре Статуса. Все прерывания имеют собственный Вектор Прерывания в Таблице Прерываний. Каждое прерывание имеет приоритет в соответствии с таблицей прерываний. Самый первый по счету вектор прерываний в Талице Прерываний, имеющий самый младший адрес, имеет наивысший приоритет.

Область памяти ввода/вывода содержит 64 адреса вывполняющих функцию Регистров Управления для периферийных устройств/функций ЦПУ и др. функций ввода/вывода. Область памяти ввода/вывода может быть адресована непосредственно или как Область Данных размещенная последовательно в Регситровом Файле от адреса 0x20 до 0x5F.

# Арифметикологичесое Устройство

Высококачественное Арифметикологическое Устройство (АЛУ) AVR работает непосредственно 32 Регистрами Общего Назначения. АЛУ выполняет операции между различными регистрами за время одного! такта. АЛУ-операции поделены на 3 основные категории: 1-арифметические, 2-логические и 3-побитовые функции. Реализация архитектуры ,также, обеспечивает возможность умножение со знаковыми и беззнаковыми числами и дробным форматом чисел. Смотри "Раздел Инструкций" там более детального описания.

# Регистр Статуса

Регистр Статуса содержит информацию о только-что выполненной арифметической инструкции. Эта информация может быть использована для того чтобы изменить ход программы выполнив условную операцию. Обратите внимание, что Регистр Статуса обновляется после каждой инструкции АЛУ, как и определено в Справочнике по Набору Команд. Это во многих случаях освобождает от надобности использования "узкоопределенных" инструкций сравнения , в результате получается эффективный по скорости выполнения и компактный по объему код программы.

При возникновении прерывания и при переходе к выполнению подпрограммы обработки прерывания Регистр Статуса должен быть предварительно сохранен куда-либо (чтобы не произошло изменение его содержимого в подпрограмме обработки прерывания), а при выходе из прерывания должен быть восстановлен обратно, если, конечно, его содержимое имеет значение.

Регистр Статуса «SREG» AVR определен следующим образом:

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| Флаги | I | T | H | S | V | N | Z | C | SREG |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**• Бит №7 - I: Глобальное Разрешение Прерываний**

Бит (флаг) "Глобального Разрешения Прерываний" должен быть установлен для того чтобы разрешить переход к обработке прерываний. Контроль над отдельными прерываниями выполняется в одельных специальных регистрах. Если Бит №7 очищен -установлен в 0- то все преывания будут запрещены независимо от индивидуальных настроек каждого из них. Бит №7 очищается аппаратно при наступлении прерывания и устанавливается в 1 командой RETI- для выхода из прерывания, чтобы разрешить последующие прерывания. Бит №7 может так же быть установлен или очищен программно по команде SEI и CLI соответственно, как описано в Справочнике по Набору Команд.

**• Бит №6 - T:Бит Загрузки и Восстановления (Буферный Бит)**

Такие инструкции Буферного Бита как BLD(загрузка бита) и BST(восстановление бита) используют Бит №6 в качестве буфера для хранения бита. Какой-нибудь бит из любого регистра может быть скопирован в Бит №6 командой BST и восстановлен оттуда командой BLD.

**• Бит №5 - H: Флаг Переноса Половины байта**

Флаг Переноса Половины H указывает на то что осуществлен перенос младшей половины байта, используется в некоторых арифметических действиях. Перенос Половины используется в BCD арифметике. См. “Описание Набора Инструкции” для детальной информации.

**• Бит №4 - S: Бит Знака, S=N** xor **V**

S-Бит всегда равен операции "Исключающее ИЛИ" между флагами N и V.

**• Бит №3 - V: Флаг Дополнения До Двух**

Этот бит подерживает арифметику с дополнением до двух.

**• Бит №2 - N: Отрицательный Флаг**

Установка этого флага говорит об отрицательном результате арифметической операции или логической.

**• Бит №1 – Z: Флаг Нуля**

Этот флага говорит, что в результате арифметической операции или логической образовался ноль.

**• Бит №0 - C: Флаг Переноса**

Установка этого флага говорит о переносе в результате арифметической операции или логической.

# Регистровый Файл

Регистровый Файл был оптимизирован специальо для RISC-инструкций AVR, что позволяет достигать заданных характеристик и гибкости. Следующие схемы ввода/вывода поддерживаются регистровым файлом:

- Один 8-битный выходной операнд и один 8-битный результирующий вход
- Два 8-битных выходных операнда и один 8-битный результирующий вход
- Два 8-битных выходных операнда и один 16-битный результирующий вход
- Один 16-битный выходной операнд и один 16-битный результирующий вход

Рисунок 4 показывает структуру 32 Регистров Общего Назначения.

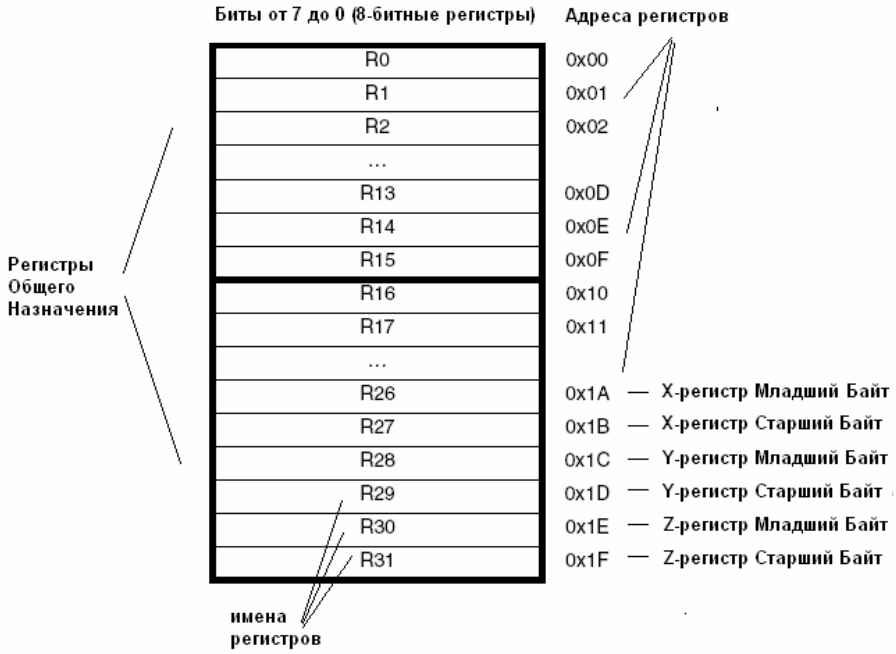



**Рисунок 4 - Структура 32 Регистров Общего Назначения**

Большинство инструкций регистрового файла имеют прямой доступ ко всем регистрам и большинство из них являются однотактовыми.

Как показано на рисунке 4, каждый регистр обозначен как адрес в памяти данных, отображены они начиная с первых 32 адресов Пространства Памяти. При том, что они физически не расположены в ОЗУ-памяти, память организована так, что обеспечивается большая гибкость доступа к регистрам, X- и Y- и Z-регистры-указатели могут указывать на индекс любого регистра в файле памяти.

# X- и Y- и Z-регистры

Регистры от R26 до R31 имеют несколько дополнительных функций помимо общих для всех остальных Регистров Общего Назначения. Эти регистры являются 16-битными указателями адреса для косвенной адресации в пространстве памяти данных. Три косвенных адресных регистра X,Y и Z определены так, как описано на рисунке5:

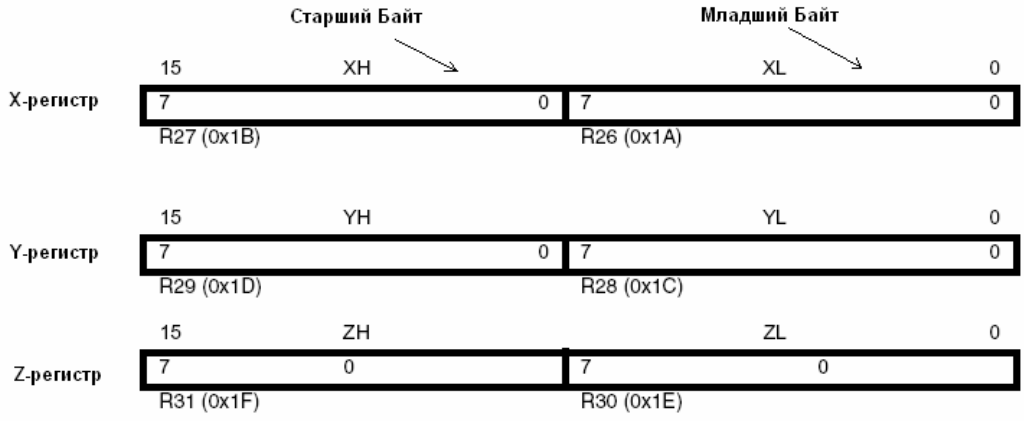



Рисунок 5 - Три косвенных адресных регистра X,Y и Z

В различных режимах адресации эти регистры адреса имеют функции как: фиксированное смещение, автоматическое возрастание{инкремент на 1}, и автоматический декремент на1. Смотри Справочник по Набору Инструкций для более подробной информации.

# Указатель Стека (счетчик программ)

"Стек" главным образом используется для запоминания(хранения) временных данных, для хранения локальных переменных и хранения адресов возврата после прерываний и вызовов подпрограмм. Регистр Указатель Стека всегда указывает на вершину (TOP) Стека. Обратите внимание, что Стек реализован нарастающим от наивысшего адреса (ячейки) памяти до самого младшего адреса. Это подразумевает, что команда PUSH (положить в Стек) уменьшает адрес Указателя Стека.

Указатель Стека указывает на участок данных ОЗУ, где и реализуется Стек подпрограмм и прерываний. Пространство Стека должно быть определено в памяти ОЗУ (обычно это конечный адрес памяти ОЗУ, если под Стек надо разрешить использовать всю память ОЗУ) программно пользователем прежде, чем произойдет любое выполнение вызова подпрограммы или будут разрешены прерывания. Указатель Стека должен быть установлен на адрес старше, чем 0x60. Указатель Стека уменьшается на единицу, когда данные кладутся в Стек по команде PUSH или уменьшается на два, когда адрес возврата записывается в Стек вызовом подпрограммы или прерыванием. Указатель Стека увеличивается на единицу, когда данные извлекаются из Стека по команде POP или увеличивается на два, когда происходит возврат из подпрограммы по команде RET или выход из прерывания по команде RETI.

Указатель Стека AVR реализован как два 8-битных регистра в пространстве ввода/вывода. Число исрользуемых битов зависит от реализации (количества памяти ОЗУ). Обратите внимание, что пространство данных (ОЗУ) в некоторых реализациях архитектуры AVR настолько мало, что можно обойтись лишь одним регистром SPL, который является младшим полубайтом регистра-указателя Стека SP. В этом случае наличие регстра SPH просто не нужно. На следующей иллюстрации показан Регистр Указателя Стека для ATtiny2313:

| номер Бита | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| | – | – | – | – | – | – | – | – | SPH |
| Имена Битов | SP7 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 | SPL |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Толькочтение | R | R | R | R | R | R | R | R | SPH |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | SPL |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SPH |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SPL |

Иллюстрация - Регистр Указателя Стека для ATtiny2313

# Время выполнения инструкций

В этой секции в общем описывается время выполнения инструкций. ЦПУ AVR -контроллера тактируется от специальной схемы генератора тактовых импульсов расположенной на чипе. Внутреннее деление тактовой частоты в этом случае не используется.

Рисунок 6 демонстрирует параллельную выборку инструкций и дальнейшее их выполнение, реализованное по гарвардской архитектуре, а также, демонстрирует концепцию быстрого доступа к Регистровому Файлу. Эта конвейерная концепция архитектуры позволяет достигать производительности до 1 миллиона инструкций на 1МГц системной тактовой синхронизации, а также, демонстрирует уникальные показатели по функциональности и стоимости прибора, по быстродействию и потреблению мощности:

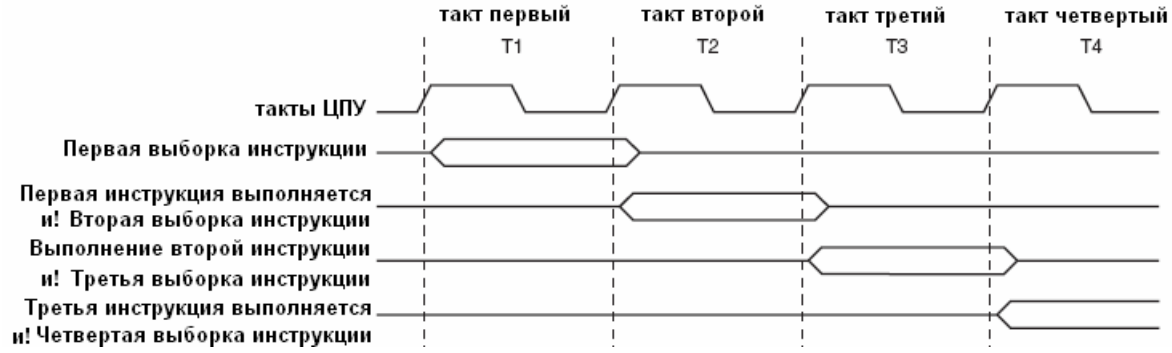



Рисунок 6 – Параллельная выборка инструкций и их дальнейшее выполнение в гарвардской архитектуре AVR

Рисунок 7 демонстрирует концепцию (понятие) о таймингах Регистрового Файла. АЛУ-операции, используя два регистровых операнда, выполняются в течении одного периода тактовой частоты, а результат операции запоминается в регистре назначения.

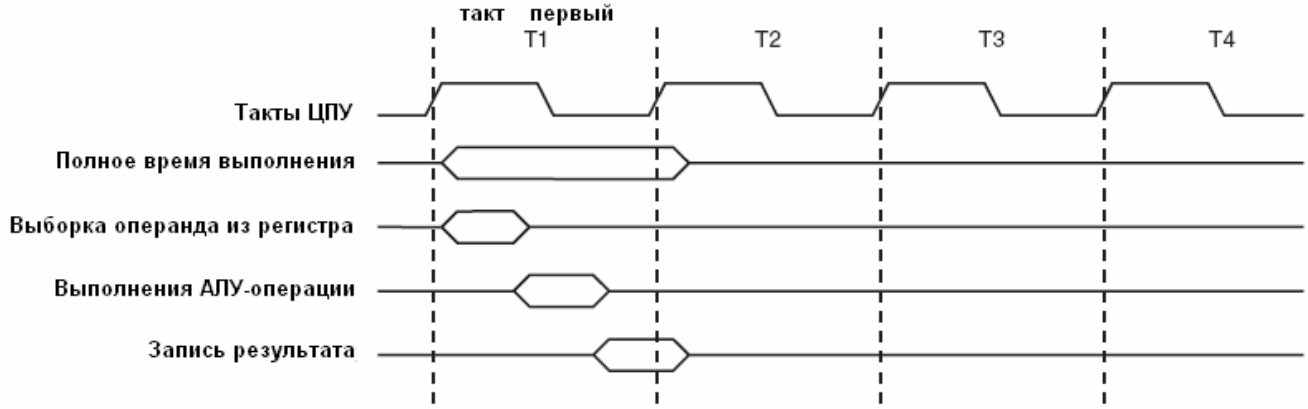



Рисунок 7 – Однотактовая АЛУ-операция

# Сброс и Обработка Прерываний

AVR имеет несколько источников прерываний. Каждое из прерываний, и отдельно Вектор Сброса, имеет собственный программный вектор перехода, расположенный в пространстве памяти. Все прерывания имеют собственный бит разрешения, в который должная быть записана логическая единица, если какое-то данное прерывание требуется разрешить. Надо помнить, что даже, если какое-то прерывание настроено и разрешено, но в Глобальный Бит Разрешения Прерываний не записана единица - то прерывание никогда не сработает.

Самые младшие адреса в памяти программ по умолчанию определены как Вектор Сброса и другие Векторы Прерывания. Полный список векторов прерываний приведен в секции "Прерывания" на странице 43. Также, этот список определяет уровень приоритета каждого прерывания. Самый младший адрес прерывания (нулевой адрес) имеет наивысший приоритет. Прерывание по сбросу (RESET) имеет наивысший приоритет, следующее прерывание по списку и с меньшим приоритетом это INT0 - "Внешний запрос на прерывание 0", и далее по списку см. стр.43.

Когда происходит прерывание, Глобальный Бит Разрешения Прерывания очищается и все другие прерывания запрещаются. Пользователь может программно записать логическую единицу в Глобальный Бит Разрешения Прерывания для разрешения остальных настроенных прерываний. В этом случае, любое возникшее прерывание сможет прервать подпрограмму обработки прерывания возникшего ранее. Глобальный Бит Разрешения Прерывания восстанавливается автоматически, когда происходит выполнение инструкции RETI - для выхода из прерывания.

Существует два основных типа прерывания. Первый тип срабатывает при установке соответствующего флага прерывания. В этом случае, Счетчик Программ "перепрыгивает" на соответствующий произошедшему прерыванию вектор (адрес), для того, что бы выполнить подпрограмму обработки прерывания, а затем аппаратно очищается соответствующий флаг прерывания. Флаг прерывания может быть очищен записью логической ЕДИНИЦЫ (именно единицы а не нуля) в бит соответствующего регистра, в котором этот флаг прерывания находится. Если условие возникновения прерывания происходит в то время, когда соответствующий бит разрешения этого прерывания очищен (установлен в ноль), флаг о возникновении прерывания все равно будет установлен (хотя прерывание в программе и не произойдет) и запомнен до того, как это прерывание будет разрешено, этот флаг может быть очищен программно. Точно так же, если произойдет одно или больше условий прерываний, в то время, как Глобальный Бит Разрешения Прерываний очищен, все равно соответствующие флаги (или один флаг) будут установлены и запомнены. А, когда Глобальный Бит Разрешения Прерываний будет установлен, произойдет обработка прерываний в соответствии с их приоритетом.

Второй тип прерываний срабатывает, когда условие возникновения прерывания присутствует в течении некоторого времени. Эти прерывания не обязаны иметь свой собственный флаг прерывания. В этом случае, если условие возникновения прерывания исчезнет до того, как это прерывание будет разрешено, тогда это прерывание никогда не будет выполнено.

Когда AVR-контроллер выходит из прерывания, он всегда возвращается на главную программу (точнее на то место кода, где прервали его выполнение) и выполняет следующую одну инструкцию прежде, чем обслужить стоящее в очереди следующее прерывание.

Обратите внимание, что содержимое Регистра Статуса автоматически не сохраняется, если происходит переход к подпрограмме обработки прерывания и не восстанавливается обратно при выходе из подпрограммы обработки прерывания. Эта работа должна выполняться программой пользователя.

Когда будет использована инструкция CLI для запрещения прерываний, прерывания будут немедленно запрещены. Ни одно прерывание не будет выполнено после инструкции CLI, даже, если условие прерывания возникает во время выполнения этой самой инструкции.

Следующий пример покажет, как можно использовать эту инструкцию, чтобы избежать прерываний во время выполнения записи в EEPROM:

Assembly Code Example

```
in  r16, SREG     ; store SREG value
cli    ; disable interrupts during timed sequence
sbi EECR, EEMPE   ; start EEPROM write
sbi EECR, EEPE
out SREG, r16     ; restore SREG value (I-bit)
```

C Code Example

```
char cSREG;
cSREG = SREG; /* store SREG value */
/* disable interrupts during timed sequence */
__disable_interrupt();
EECR |= (1<<EEMPE); /* start EEPROM write */
EECR |= (1<<EEPE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

Когда используется инструкция SEI для разрешения прерываний, то инструкции, следующие после команды SEI, будут выполняться до наступления какого-либо прерывания, как показано на этом примере:

Assembly Code Example

```
sei  ; set Global Interrupt Enable
sleep; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)
```

C Code Example

```
__enable_interrupt(); /* set Global Interrupt Enable */
__sleep(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

# Время Реакции на Прерывания

Время реакции на все разрешенные прерывания в контроллере составляет минимум 4 такта. Спустя 4 такта происходит переход на вектор возникшего прерывания и дальнейший вызов подпрограммы обработки прерывания. В течении перилда времени этих 4 тактов происходит сохранение Счетчика Программ в Стеке. Затем происходит переход вектора программы на метку подпрограммы для обработки прерывания, это занимат 3 такта. Если преывание происходит во время выполнения многотактовой инструкции то сначала происходит завершение выполнения этой инструкции, а потом обработка прерывания. Если прерывание происходит, когда ЦПУ (МЦУ) находится в режиме сна, то время реакции на прерывание увеличивается на 4 такта. Такое увеличение времени реакции происходит из-зи добавления времени тебуемого на выход из выбранного режима сна.

На выход из подпрограммы обработки прерывания тоже требуется 4 такта. В течении врнмени этих 4 тактов Счетчик Программ (двух-байтный) выталкивается -восстанавливается- из Стека, Указатель Стека увеличивается на 2 и происходит установка бита I в регистре состояния SREG.

# Память AVR ATtiny2313

Эта секция описывает разные виды памяти в ATtiny2313. AVR архитектура имеет два главных пространства памяти, Память Данных и Память Программ. В дополнение кэтому, ATtiny2313 имеет EEPROM-память (ПЗУ) для хранения данных. Эти три пространства памяти расположены последовательно.

# Встроенная программируемая ФЛЭШ-память

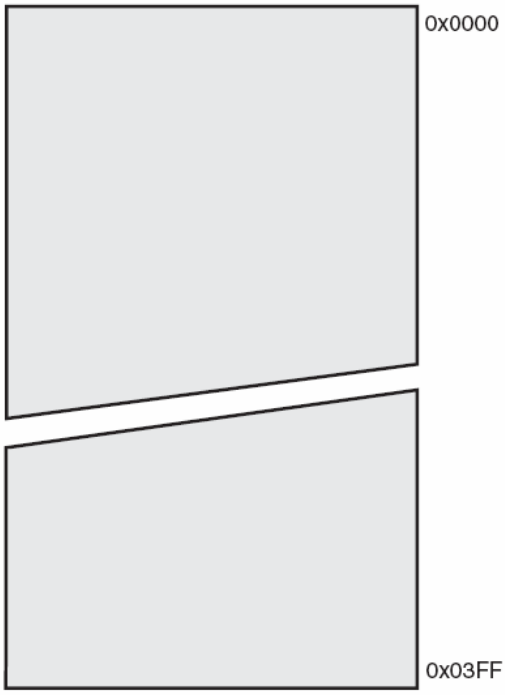
ATtiny2313 содержит встроенную ФЛЭШ-память для хранения программы. Так как все AVR-инструкции являются 16- или 32-битными то ФЛЭШ-память организована как таблица 1К x 16.

ФЛЭШ-память может быть перезаписана мнимум 10 000 раз, имеются в виду циклы записи и стирания. Счетчик Программ (PC) у ATtiny2313 является 10-битным, так что может адресовать ровно 1К (1 КилоБайт) пространства памяти. Раздел "Программирование Памяти" на стр157 содержит детальное описание как программировать (прошивать) ФЛЭШ-память через SPI-интерфейс.

Таблица констант может быть расположена в пределах пространства всей памяти (смотрите описание инструкции для Загрузки Памяти Программ- LPM- Load Programm Memory)

Временные диаграммы для выборки инструкций и для выполнения инструкций представлены на стр10.

**Память Программ**



# Память данных SRAM (ОЗУ)

Рисунок 9 показывает как организована память данных SRAM.

Самые младшие 224 адреса расположения данных в памяти данных это Файл Регистров, память ввода/вывода, Расширенная память ввода/вывода и внутренняя память данных SRAM. Первые 32 адреса - это и есть Файл Регистров (Регистровый Файл), следующие 64 адреса - это стандартная память ввода/вывода, а следующие 128 адресов это уже внутренняя память данных - ОЗУ (SRAM).

Пять различных режимов адресации памяти данных представляют собой следующее:

1-Прямой Доступ,

2-Косвенная Адресация со Смещением,

3-Косвенная Адресация (Простая),

4-Косвенная Адресация с предварительным декрементом (предекремент),

5-Косвенная Адресация с последующим инкрементом (постинкремент).

В Регистровом Файле регистры с R26 до R31 являются регистрами косвенной адресации (косвенные указатели адреса).

Прямая адресация может адресовать полностью все пространство памяти.

Косвенная Адресация со смещением может адресовать 63 адреса используя регистры Y или Z.

Когда используется режим Косвенной Адресации с Предекрементом или Постинкрементом то регистры X или Y или Z декрементируются или инкрементируются.

32 Регистра Общего Назначения, 64 регистра ввода/вывода и 128 байт внутренней памяти ОЗУ в контроллере, все это становится доступным путем использования всех пяти режимов адресации.

Регистровый Файл описан на стр8.


Рисунок 9 – Карта Памяти Данных

# Время Доступа к Памяти Данных

Эта секция описывает концепцию времени доступа к Внутренней Памяти Данных. Время доступа к внутренней ОЗУ занимает 2 такта ЦПУ, как показано на Рисунке 10:

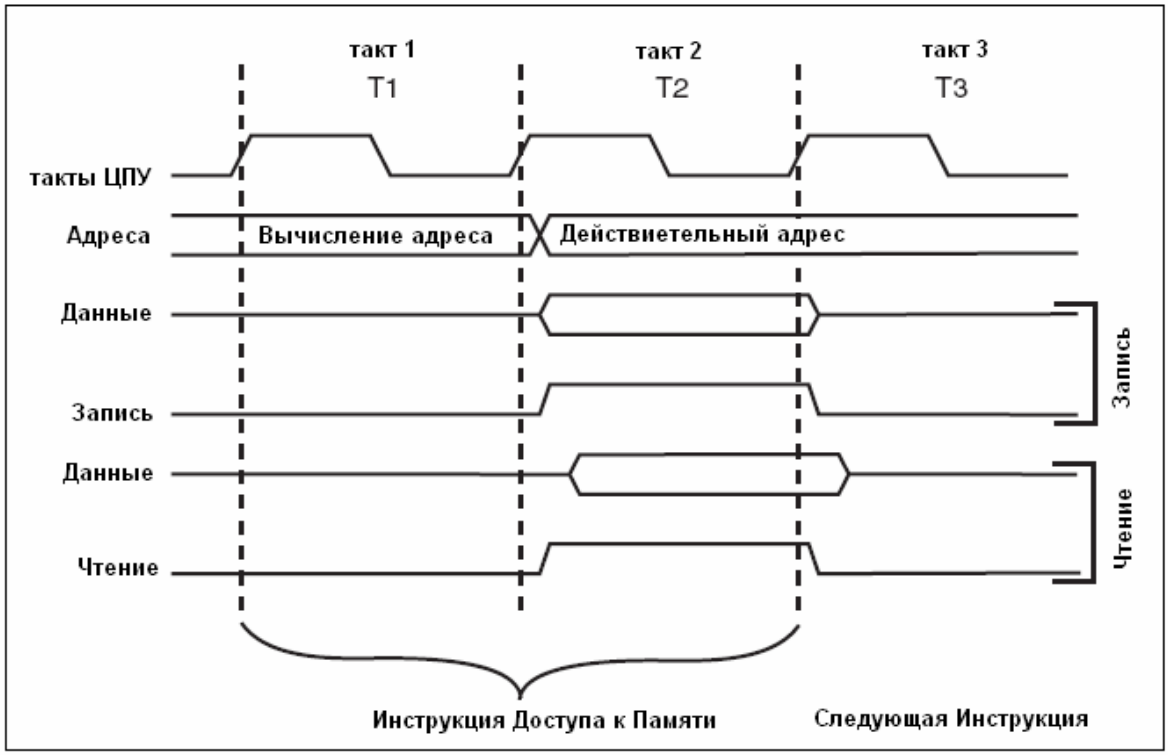



Рисунок 10 – Временные диаграммы доступа к данным в ОЗУ встроенному на кристалл

# Память Данных EEPROM (ППЗУ)

ATtiny2313 содержит 128 Байт EEPROM (ППЗУ)-памяти. Эта память организована как отдельное пространство данных, каждый один байт может быть прочитан и записан. EEPROM (ППЗУ) может быть записана и стерта в сумме как минимум 100 000 раз. Работа ЦПУ с EEPROM объяснена в описании Регистра Адреса EEPROM, Регистра Данных EEPROM и Регистра Управления EEPROM. За подробной информацией по последовательной загрузке данных в EEPROM обратитесь к стр. 171.

# Доступ на чтение/запись EEPROM

Регистр Доступа к EEPROM расположен в пространстве ввода/вывода. Время доступа к записи EEPROM дано в Таблице 1. Эта самоопределяющая временная функция и позволяет пользователю программно определять, когда же можно будет записывать следующий байт. Если программа пользователя содержит инструкции записи, тогда должны быть предприняты некоторые предосторожности. В источниках питания содержащих фильтры возможно медленное изменение напряжения Vcc (+питания) вверх или вниз относительно земли. По этой причине устройство может работать некоторое время с пониженным напряжением питания, меньшим, чем положено, для данной системной частоты. Смотри "Предотвращение Искажения Данных EEPROM" на стр. 19 для подробной информации о том, как избежать проблем в этой ситуации. Для того, чтобы избежать неумышленную запись в EEPROM, должны быть соблюдены определенные процедуры. Обратитесь за более детальной информацией по этому поводу к описанию Регистра Контроля EEPROM. Когда происходит чтение EEPROM, ЦПУ приостанавливается на 4 такта до того, как начать выполнение следующей инструкции. После записи EEPROM ЦПУ приостанавливается на 2 такта, прежде, чем начать выполнение следующей инструкции.

# Регистр Адреса EEPROM «EEAR»

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | – | EEAR6 | EEAR5 | EEAR4 | EEAR3 | EEAR2 | EEAR1 | EEAR0 | EEAR |
| Чтение/Запись | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Начальное Значение | 0 | X | X | X | X | X | X | X | |

**• Бит №7 - Res: Зарезервированный Бит**

Этот бит зарезервирован в ATtiny2313 и всегда читается как ноль.

**• Биты №6.. №0 - EEAR6.. 0: Адреса EEPROM**

Регистр адреса - EEAR - определяет адрес байта данных в 128 байтном пространстве EEPROM. Байты данных EEPROM адресуются линейно от 0 адреса до 127. Начальные значения битов в Регистре EEAR не определены. Надлежащие значения этого регистра должны быть записаны, до того, как EEPROM будет использоваться.

# Регистр Данных EEPROM «EEDR»

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | MSB | | | | | | | LSB | EEDR |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Начальное Значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**• Биты №7.. №0 - EEDR7.. 0: EEPROM Данные**

Для записи в EEPROM, Регистр Данных EEDR должен содержать данные, которые будут записываться в EEPROM, по адресу, определенному в Регистре Адреса EEAR. Для операции чтения EEPROM, Регистр Данных EEDR должен содержать данные прочитанные из EEPROM по адресу, определенному в Регистре Адреса EEAR.

# Регистр Управления EEPROM «EECR»

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | – | – | EEPM1 | EEPM0 | EERIE | EEMPE | EEPE | EERE | EECR |
| Чтение/Запись | R | R | R/W | R/W | R/W | R/W | R/W | R/W | |
| Начальное Значение | 0 | 0 | X | X | 0 | 0 | X | 0 | |

**• Биты №7.. №6 - Res: Зарезервированные Биты**

Эти биты зарезервированы в ATtiny2313 и всегда читаются как ноль.

**• Биты №5, №4 - EEPM1 и EEPM0: Программирование Битов Режима EEPROM**

Установка битов выбора режима EEPROM определяет какое действие выполнит программа, когда в бит EEPE будет загружена 1. Это позволяет программировать данные в автоматическом режиме (стирание старых данных и записывание новых данных) или выполнять от отдельно операции Стирания и Записи. Время программирования данных для различных режимов показано в таблице 1. Пока бит EEPE содержит 1, любая запись в бит EEPMn будет игнорироваться. При сбросе контроллера биты EEPMn будут сброшены в 0b00, если EEPROM в этот момент программировалась.

Таблица 1 - Биты выбора режимов EEPROM

| бит EEPM1 | бит EEPM0 | Время программирования | Операции |
|---|---|---|---|
| 0 | 0 | 3.4 ms | Стирание и Запись в одну операцию (автоматическая операция) |
| 0 | 1 | 1.8 ms | Только Стирание |
| 1 | 0 | 1.8 ms | Только Запись |
| 1 | 1 | – | Зарезервировано для будущего использования (еще не придумали, что с этим делать) |

**• Бит №3 - EERIE: Прерывание от EEPROM по Состоянию "готово к записи"**

В бит EERIE нужно записать 1, чтобы разрешить прерывание (и переход к к вектору EE READY), если так же установлен в 1 бит I в Регистре Состояния SREG. Запись в EERIE нуля запрещает прерывание от EEPROM. Если в EERIE находится единица, то прерывание будет происходить постоянно, когда EEPROM будет готова к записи.

**• Бит №2 - EEMPE: Мастер Программного Разрешения EEPROM**

Бит EEMPE определяет будет ли эффект от записанного бита EEPE. Если в бит EEMPE записать 1 и потом, не позже, через 4 такта, записать 1 в бит EEPE, то начнется программирование байта данных в EEPROM по заданному адресу. Если в EEMPE ноль то установка в 1 EEPE не дает эффекта. Если в EEMPE записать 1 программно, то спустя 4 такта этот бит будет очищен в ноль аппаратно.

**• Бит №1 - EEPE: Программное Разрешение в EEPROM**

Бит EEPE это бит Программного Разрешения Сигнала идущего в EEPROM, программирует разрешение для сигнала (байта) идущего в EEPROM. Когда EEPE записан 1-ей то EEPROM будет запрограммирована согласно установкам битов EEPMn. EEMPE бит должен уже содержать единицу еще до того, как единица будет записана в бит EEPE, иначе запись EEPROM не будет осуществлена. Когда отведенное время на доступ к записи истечет, бит EEPE будет очищен аппаратно. Когда будет установлен бит EEPE, ЦПУ приостановится на два цикла до того, как следующая инструкция будет выполнена.

**• Бит №0 - EERE: Разрешение Чтения EEPROM**

Бит EERE - это бит Разрешения Чтения Сигнала (байта) из EEPROM. Бит EERE является стробирующим сигналом для EEPROM. После того, как будет записан корректный адрес в Регистр Адреса - EEAR - тогда можно будет записывать 1 в бит EERE для вызова чтения из EEPROM. Доступ к EEPROM будет осуществлен за одну инструкцию и требуемые данные будут тут же доступными. Когда EEPROM данные будут прочитаны, ЦПУ приостановится на 4 такта прежде, чем начать выполнение следующей инструкции. Пользователь должен программно опрашивать бит EEPE до того, как начать операцию чтения. Если в данный момент времени операция записи EEPROM будет активна, то нельзя осуществлять чтение EEPROM и изменение содержимого Регистра Адреса EEAR.

# Автоматическое Программирование Байта

Наиболее простым из всех является режим Автоматического Программирования Байта. Перед тем как осуществить запись в EEPROM пользователь должен сначала записать адрес в Регистр Адреса EEAR и данные в Регистр Данных EEDR. Если биты EEPMn являются нулевыми, то запись 1 в бит EEPE (не позднее, чем в течении 4 циклов после записи 1 в EEMPE) приведет к запуску операций стирания и записи. Циклы стирания и записи будут выполнены как одна операция, полное время такого программирования памяти приведено в Таблице 1 на стр.16.
Бит EEPE остается установленным до того, пока циклы стирания и записи не завершатся. Пока устройство занято программированием EEPROM, невозможно с EEPROM выполнять какие-либо другие операции.

# Раздельное Программирование Байтов

Также, возможно выполнять две разные операции Записи и Стирания. Это может понадобиться в приложениях требующих быстрого доступа к памяти за некоторые ограниченные периоды времени (например, при падающем напряжении). Чтобы воспользоваться преимуществами такого метода, нужно помнить, что ячейка памяти по тому адресу, куда будет производиться запись данных, должна быть предварительно стерта. Стирание нужно производить, когда напряжение является достаточно высоким для нормальной работы.

# Стирание

Чтобы произвести стирание байта, прежде всего надо записать в Регистр Адреса EEAR адрес стираемого байта. Если выбран режим "Только Стирание", как показано в Таблице 1 - 0b01 для EEPMn, то запись EEPE (в течении 4 тактов, после записи EEMPE) вызовет операцию стирания. Бит EEPE останется установленным в 1 до завершения операции стирания. Пока устройство занято стиранием EEPROM, невозможно с EEPROM выполнять какие-либо другие операции.

# Запись

Чтобы произвести запись байта, прежде всего надо записать в Регистр Адреса EEAR адрес записываемого байта. Если выбран режим "Только Запись", как показано в Таблице 1 - 0b10 для EEPMn, то запись 1 в EEPE (в течении 4 тактов, после записи EEMPE) вызовет операцию записи байта. Бит EEPE останется установленным в 1 до завершения операции записи. Если ячейка памяти, куда производится запись не была стерта перед записью, то данные хранимые в ней будут затерты новыми. Пока устройство занято записью EEPROM, невозможно с EEPROM выполнять какие-либо другие операции.

Калибровка Генератора происходит примерно за то же время ,что и доступ к EEPROM. Убедитесь, что частота Генератора находится в пределах требуемой в описании "Регистра Калибровки Генератора - OSCCAL" на стр. 25.

Следующий пример кода покажет, как на ассемблере и на С производить запись в EEPROM. В примерах принято, что прерывания были отключены путем очистки бита I в Регистре Состояния SREG, так что прерывания не сработают во время выполнения функций в примерах:

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic EECR,EEPE
  rjmp EEPROM_write
  ; Set up address (r17) in address register
  out EEAR, r17
  ; Write data (r16) to data register
  out EEDR,r16
  ; Write logical one to EEMPE
  sbi EECR,EEMPE
  ; Start eeprom write by setting EEPE
  sbi EECR,EEPE
  ret
```

C Code Example

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEPE))
    ;
  /* Set up address and data registers */
  EEAR = uiAddress;
  EEDR = ucData;
  /* Write logical one to EEMPE */
  EECR |= (1<<EEMPE);
  /* Start eeprom write by setting EEPE */
  EECR |= (1<<EEPE);
}
```

Следующий пример кода покажет, как на ассемблере и на С производить чтение из EEPROM. В примерах принято, что прерывания были отключены путем очистки бита I в Регистре Состояния SREG, так что прерывания не сработают во время выполнения функций в примерах:

Assembly Code Example

```
EEPROM_read:
  ; Wait for completion of previous write
  sbic EECR,EEWE
  rjmp EEPROM_read
  ; Set up address (r17) in address register
  out  EEAR, r17
  ; Start eeprom read by writing EERE
  sbi  EECR,EERE
  ; Read data from data register
  in   r16,EEDR
  ret
```

C Code Example

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEWE))
    ;
  /* Set up address register */
  EEAR = uiAddress;
  /* Start eeprom read by writing EERE */
  EECR |= (1<<EERE);
  /* Return data from data register */
  return EEDR;
}
```

# Предотвращение Искажения данных EEPROM

В течении периода времени низкого напряжения питания Vcc, EEPROM данные могут быть искажены, потому, что ЦПУ и EEPROM не могут корректно работать в таких условиях. Эта проблема является такой же, как у плат-приложений использующих EEPROM и решения этой проблемы здесь могут быть такими же. Искажение данных EEPROM при низком напряжении может произойти по двум причинам. Первая: последовательная запись в EEPROM требует определенного минимального напряжения для корректной работы. Вторая причина: ЦПУ может сам непосредственно выполнять некорректно операции, если напряжение источника питания слишком низкое.

Искажения данных EEPROM можно легко избежать, следуя следующим проектным рекомендациям:

Удерживать выход RESET в низком состоянии в течении времени пока напряжение неудовлетворительное. Это может быть сделано разрешив работу Brown-out Detector (BOD) - детектора низкого напряжения. Если определяемый уровень напряжения внутренним BOD-детектором не соответствует заданному уровню, то сработает Схема Защиты (сброса).

Если Сброс Схемой Защиты должен произойти во время операции записи в EEPROM, то будет обеспечено завершение этой операции, но при условии, что для этого будет достаточно напряжения.

# Память Ввода/Вывода (в/в)

Определение пространства в/в для ATtiny2313 приведено на стр. 209 в разделе "Резюме Регистра".

Все виды пространств в/в в ATtiny2313 и периферийные устройства расположены в общем пространстве в/в. Любое место пространства в/в может быть доступно через инструкции LD/LDS/LDD и ST/STS/STD при передачи данных между 32 Регистрами Общего Назначения и самим пространством в/в. Регистры в/в, чьи адреса располагаются в пределах диапазона 0x00 - 0xF1, побитно доступны через команды SBI и CBI. В этих регистрах значения каждого бита может быть проверено командами SBIS и SBIC. За подробной информацией об инструкциях обращайтесь к соответствующим разделам. Определенные команды для пространства в/в, такие как IN и OUT, должны использоваться в диапазоне адресов 0x00-0x3F. Когда Регистры пространства в/в адресуются как пространство данных, посредством коаманд LD и ST, то к адресам этих Регистров должно быть прибавлено значение 0x20.

Зарезервированные адреса памяти пространства в/в никогда не могут быть перезаписаны. Некоторые флаги (биты) в Регистре Состояния SREG очищаются записью в них именно логической ЕДИНИЦЫ а не нуля.

Обратите внимание, что в отличие от других инструкций контроллера, CBI и SBI инструкции оперируют с определенными битами и поэтому они могут быть использованы в регистрах для определения отдельного флага.

Команды CBI и SBI работают только в определенном диапазоне адресов: от 0x00 до 0x1F.

Регистры Управления пространством в/в и Периферией объяснены в следующем разделе.

# Регистры в/в Общего Назначения

ATtiny2313 содержит три Регистра Ввода/Вывода Общего Назначения. Эти регистры могут быть использованы для запоминания любой информации и особенно они полезны для запоминания Глобальных Переменных и Флагов Состояния. Регистры Ввода/Вывода Общего Назначения находятся в диапазоне адресов 0x00 - 0x1F и непосредственно побитно доступны через инструкции SBI, CBI, SBIS, и SBIC .

# Регистр Ввода/Вывода Общего Назначения 2 – GPIOR2 (General Purpose I/O Register)

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | MSB | | | | | | | LSB | GPIOR2 |
| Чтение/запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

# Регистр Ввода/Вывода Общего Назначения 1 – GPIOR1 (General Purpose I/O Register)

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | MSB | | | | | | | LSB | GPIOR1 |
| Чтение/запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

# Регистр Ввода/Вывода Общего Назначения 0 – GPIOR0 (General Purpose I/O Register)

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | MSB | | | | | | | LSB | GPIOR0 |
| Чтение/запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

# Системный Тактовый Генератор и его Настройка

## Описание Системного Генератора

На Рисунке 11 представлена структурная схема Тактового Генератора и его описание. Все устройства нуждающиеся в Тактовом Генераторе будут неактивны, если последний тоже неактивен. Чтобы уменьшить потребление мощности, тактовый модуль может быть остановлен в нескольких на выбор режимах сна, описанных на странице 29 в разделе "Управление Энергопотреблением и Режимы Сна". Тактовый Генератор детально представлен ниже:

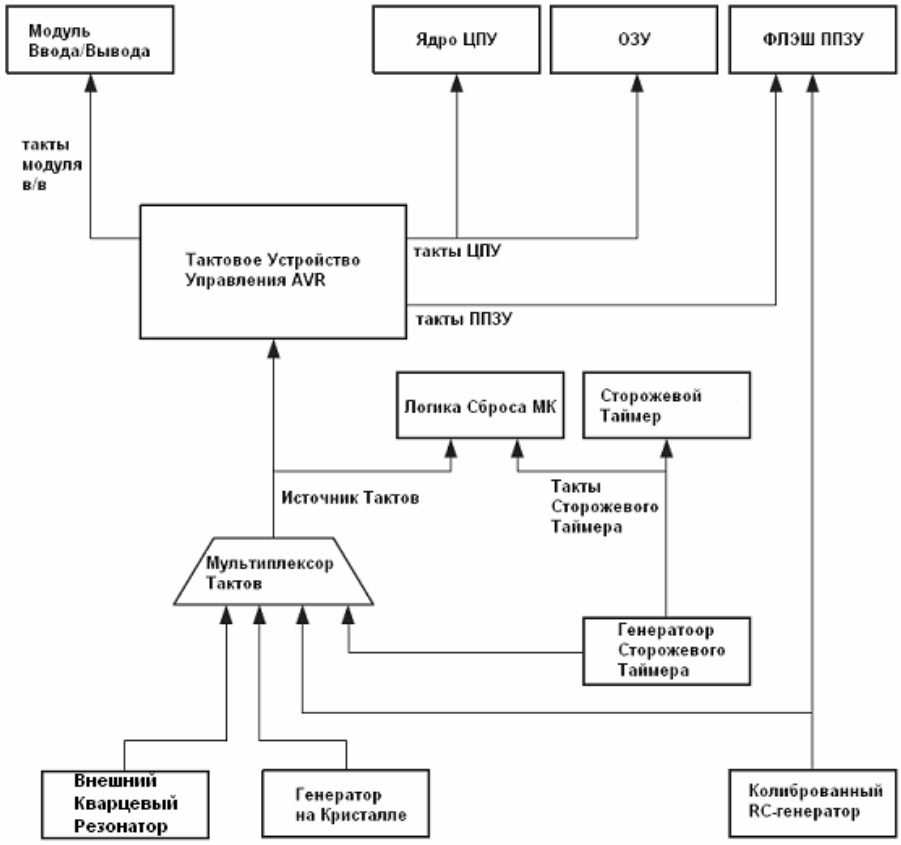



Рисунок 11 - Структурная схема Тактового Генератора

# Такты ЦПУ

Такты ЦПУ направлены на те части системы (модули), которые работают синхронно с ядром AVR. К примеру, такими модулями могут быть Регистры Общего Назначения, Регистр Статуса и память данных, содержащая Указатель Стека. Остановка тактов ЦПУ запрещает работу ядра и прекращает выполнение общих операций и вычисления.

# Такты пространства в/в

Такты пространства в/в используются большинством модулей в/в, такими, как Таймеры/Счетчики и интерфейс USART. Такты пространства в/в так же используются модулем Внешних Прерываний, но не теми внешними прерываниями, которые используют асинхронную логику, позволяя таким прерываниям срабатывать даже, если такты пространства в/в остановлены. Также, обратите внимание, что условие начального обнаружения в USI-модуле является асинхронным, когда такты в/в остановлены; это условие в USI-модуле запускается при переходе в спящий режим.

# Такты ППЗУ

Операции тактового управления ППЗУ определяют интерфейс ППЗУ. Такты ППЗУ одновременно используются с тактами ЦПУ.

# Источник Тактирования

Контроллер имеет следующие настройки источника тактовых импульсов, выбираемые прошивкой Fuse-битов (битов-перемычек), как показано ниже. Сигналы из выбранного источника идут на вход Тактового Генератора AVR и затем поступают на соответствующие модули.

**Таблица 2 Выбор схемы тактирования**

| Настройка тактрующего устройства | CKSEL3..0 |
|---|---|
| Внешнее тактирование | 0000 |
| Калиброванный внутренний RC-генератор на 4 МГц | 0010 |
| Калиброванный внутренний RC-генератор на 8 МГц | 0100 |
| Генератор Сторожевого Таймера на 128 КГц | 0110 |
| Внешний Кварцевый Резонатор | 1000 - 1111 |
| Зарезервировано | 0001/0011/0101/0111 |

**Примечание:**
**1. Для всех перемычек состояние:**

**"1"-означает НЕзапрограммировано, а**
**"0"-ЗАпрограммировано.**

В следующих разделах описаны опции каждого способа тактирования. Когда ЦПУ пробуждается ото сна, то начинает стартовать выбранный источник тактирования обеспечивая стабильную частоту еще до начала выполнения инструкций. Когда ЦПУ стартует после Сброса то это происходит с задержкой, позволяя источнику тактирования стабилизировать свою работу, еще до начала нормального функционирования контроллера. Генератор Сторожевого Таймера как раз и используется для подсчета промежутка времени задержки при старте контроллера. В таблице 3 показаны номера тактовых циклов Генератора Сторожевого Таймера (WDT), используемых при подсчетах времени. Частота Генератора Сторожевого Таймера зависит от приложенного напряжения, об этом говорится на стр. 179 в разделе "Типичные Характеристики МК ATtiny2313"

**Таблица 3 – Количество тактов Генератора Сторожевого Таймера**

| Типичное Конечное время (при Vcc=5V) | Типичное Конечное время (при Vcc=3V) | Номер Такта (конечное число) |
|---|---|---|
| 4,1 мс | 4,3 мс | 4К (4096) |
| 65 мс | 69 мс | 64К (65536) |

# Источник тактирования по-умолчанию

Контроллер выпускается с такими запрограммированными битами по-умолчанию:
CKSEL = “0010”, SUT = “10”, и CKDIV8.

Внутренний RC-генератор по-умолчанию настроен на наибольшее время запуска с предварительным делением частоты на 8. Установки по-умолчанию гарантируют, что каждый пользователь сможет в своих проектах настраивать источник тактирования при ВнутриСистемном или Параллельном программировании.

# Генератор на Резонаторе

XTAL1 и XTAL2 это вход и, соответственно, выход инвертирующего усилителя, который может быть сконфигурирован пользователем как внутренний генератор, как показано на Рисунке 12 на стр. 23.Также, можно использовать Кварцевый резонатор или Керамический.

Емкости C1 и C2 всегда будут эквивалентными, т.е, одинаковыми при использовании резонатора любого вида. Оптимальное значение емкостей определяется паразитными емкостями и электромагнитными шумами внешней среды. Несколько начальных рекомендаций по выбору емкостей, при использовании резонаторов, дано в Таблице 4 на стр. 23. Для керамических резонаторов значения емкости определяется опираясь на информацию производителя резонатора.

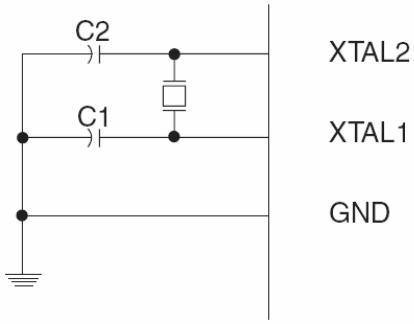



Рисунок 12 – Схема подключения Резонатора

Генератор может работать в трех разных режимах, каждый оптимизирован под определенный диапазон частот. Режим работы выбирается fuse-битами (битами-перемычками) CKSEL3..1, как показано в Таблице 4:

**Таблица 4 – Режимы работы Кварцевого Генератора**

| биты CKSEL3..1 | Диапазон частот [1] (МГц) | Рекомендованный диапазон для емкостей C1 и C2 с номиналами (пФ) |
|---|---|---|
| 100[2] | 0.4 - 0.9 | – |
| 101 | 0.9 - 3.0 | 12 - 22 |
| 110 | 3.0 - 8.0 | 12 - 22 |
| 111 | 8.0 - | 12 - 22 |

**Примечание:**
1. **Предварительные значения диапазона частот**
2. **Эта настройка не применима для Кварцевых резонаторов - только для Керамических.**

Биты-перемычки CKSEL0, вместе с битами SUT1..0, выбирают время запуска, как указано в Таблице 5:

**Таблица 5 – Время запуска Тактового Генератора с Кварцевым резонатором или Керамическим**

| биты CKSEL0 | биты SUT1..0 | Время запуска из Холстого режима и Экономичного | Дополнительная задержка после СБРОСА (Vcc=5V) | Рекомендовано использовать когда: |
|---|---|---|---|---|
| 0 | 00 | 258 CK(1) | 14CK + 4.1 ms | керамический резонатор быстронарастающее напряжение |
| 0 | 01 | 258 CK(1) | 14CK + 65 ms | керамический резонатор и напряжение нарастает медленно |
| 0 | 10 | 1K CK(2) | 14CK | керамичекий резонатор и включен Brown-out Detector |
| 0 | 11 | 1K CK(2) | 14CK + 4.1 ms | керамический резонатор быстронарастающее напряжение |
| 1 | 00 | 1K CK(2) | 14CK + 65 ms | керамический резонатор и напряжение нарастает медленно |
| 1 | 01 | 16K CK | 14CK | Кварцевый резонатор и включена BOD-схема |
| 1 | 10 | 16K CK | 14CK + 4.1 ms | Кварцевый резонатор и быстро нарастающее напряжение Vcc |
| 1 | 11 | 16K CK | 14CK + 65 ms | Кварцевый резонатор и медленно нарастающее напряжение питания |

Примечание:

1. Эта опция используется только тогда, когда устройство не работает на максимальной или близкой к ней частоте и, когда в приложении не требуется стабильная частота при старте.
2. Эта опция предназначена для использования Керамического резонатора и гарантирует стабильную частоту при старте. Она, также, может быть использована и с Кварцевым резонатором, но, если устройство не работает на максимальной частоте и не требуется стабильность частоты при старте.

# Калиброванный Внутренний RC-генератор

Калиброванный Внутренний RC-генератор обеспечивает тактовую частоту 8.0 МГц. Это номинальное значение частоты при Vcc=3V и 25*C. Если частота 8.0 МГц это слишком высокая частота при заданном напряжении, как указано в спецификации на устройство, то необходимо перепрограммировать бит-перемычку CKDIV8 для того, чтобы поделить эту частоту на 8. Контроллер по-умолчанию выпускается с уже установленным битом CKDIV8. Такая частота может быть выбрана в качестве системной, путем программирования CKSEL-битов, как показано в Таблице 6. В таком случае, контроллер сможет работать без какого-либо внешнего резонатора. После СБРОСА контроллера, происходит аппаратная загрузка калибрующего байта в Регистр OSCCAL и таким образом автоматически калибруется RC-генератор. При напряжении питания 3 В и температуре 25* С, такая калибровка обеспечивает частоту с ошибкой в 10% от номинальной. Если использовать методы калибрации, которые описаны и доступны на html-странице www.atmel.com/avr, то возможно достигнуть ошибки установки частоты до 2% от номинальной при любом напряжении Vcc и Температуре. Если RC-генератор используется как источник системной частоты, то для Сторожевого Таймера по-прежнему будет использоваться свой собственный генератор, а также, и для формирования задержки после сброса контроллера. Для более подробной информации о программировании калибрующих значений обратитесь к разделу "Калибрующий Байт" на стр. 159.

Таблица 6 – Режимы внутреннего калиброванного RC-генератора

| CKSEL3..0 | Номинальная частоа |
|---|---|
| 0010 - 0011 | 4.0 MHz(1) |
| 0100 - 0101 | 8.0 MHz |

Примечание: 1. МК выпускается по-умолчанию с такими настройками

При выбор этого Генератора, время запуска определяется SUT-перемычками, как показано в таблице 7:

Таблица 7. Время запуска для внутреннего калиброванного RC-генератора (выбранного в качестве системного генератора)

| SUT1..0 | Время запуска при выходе из PowerDown-Экономичного режима и Холостого (PowerSave) | Добавляемая задержка при старте после Сброса (Vcc=5 V) | Рекомендовано использовать при: |
|---|---|---|---|
| 00 | 6 CK | 14CK | включенной BOD-схеме |
| 01 | 6 CK | 14CK + 4.1 ms | Быстро нарастающем напряжении питания |
| 10(1) | 6 CK | 14CK + 65 ms | Медленно нарастающем напряжении питания |
| 11 | **Заблокировано** | | |

**Примечание: 1. Устройство выпускается при таких выбранных настроках**

# Регистр OSCCAL для калибровки Генератора

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | – | CAL6 | CAL5 | CAL4 | CAL3 | CAL2 | CAL1 | CAL0 | OSCCAL |
| Чтение/Запись | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |

**• Биты 6.. 0 - CAL6.. 0: Значения калибрующие Генератор**

Значения, записанные по этому адресу, могут настроить Внутренний Генератор и уменьшить нестабильность частоты. Это делается автоматически при перезагрузке Чипа. Когда в регистре OSSCAL нули, это означает, что выбрана самая низкая частота. Запись в этот регистр ненулевых значений, приведет к повышению частоты Внутреннего Генератора. Запись в Регистр значения 0x7F настраивает Генератор на максимально доступную частоту. Внутренний Генератор может использоваться при обращении к EEPROM и FLASH памяти. При записи EEPROM или FLASH, ошибка в стабильности Внутреннего Генератора не должна превышать 10%. В противном случае запись может быть неудачной. Заметьте, что этот Генератор предназначен для настройки на частоту 8 или 4 МГц. Настройка на другие частоты не гарантирована, как показано в Таблице 8.

Избегайте больших отклонений частоты Внутреннего Генератора от номинальной для обеспечения стабильной работы Чипа. Отклонение от номинальной частоты больше, чем на 2% за время между двумя тактовыми периодами, может привести к непредсказуемым последствиям. Изменения в регистре OSCCAL не должны превышать числа 0x20 при каждой калибровке.

Таблица 8. Диапазон частот Внутреннего RC-генератора

| Значения регистра OSCCAL | Минимальная частота в процентах от номинальной частоты | Максимальная частота в процентах от максимальной частоы |
|---|---|---|
| 0x00 | 50% | 100% |
| 0x3F | 75% | 150% |
| 0x7F | 100% | 200% |

# Внешний тактовый Генератор

Для тактировать Чип от Внешнего тактового Генератора, нужно воспользоваться выводом XTAL1 так, как показано на Рисунке 13. Чтобы Чип смог запуститься от Внешнего Генератора, нужно позаботиться о том, чтобы Биты-перемычки CKSEL были запрограммирован в нули - "0000".

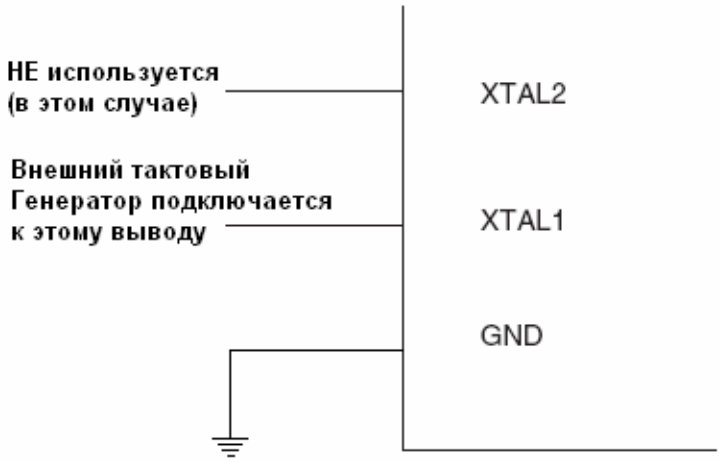



Рисунок 13. Подключение Внешнего Тактового Генератора

Если выбран этот источник тактирования, то время запуска определяется SUT-битами перемычками, как показано в Таблице 10:

**Таблица 9. Тактовая частота Чипа**

| CKSEL3..0 | Диапазон частот |
|---|---|
| 0000 - 0001 | 0 - 16 MHz |

**Таблица10. Время запуска с Внешним тактовым Генератором**

| SUT1..0` | Время запуска при выходе из холостого режима PowerSave(idle) и экономичного PowerDown | Добавляемая задержка после Сброса (Vcc=5V) | Рекомендовано использовать при: |
|---|---|---|---|
| 00 | 6CK | 14CK | Включенной BOD-схеме |
| 01 | 6CK | 14CK+4.1мСек | Быстро нарастающем Vcc |
| 10 | 6CK | 14CK+65мСек | Медленно нарастающем Vcc |
| 11 | зарезервировано | зарезервировано | зарезервировано |

Использование Внешнего Генератора должно обеспечивать отсутствие внезапных изменений в тактовой частоте, чтобы работа Чипа была стабильной. Изменение в частоте более чем на 2% между двумя соседними тактовыми периодами может привести к непредсказуемым событиям. Невыполнение этих требований, гарантирует, что контроллер будет находиться в состоянии Сброса, пока частота не стабилизируется.

Заметьте, что Системный Тактовый Делитель может быть использован для того, чтобы изменять время запуска Внутреннего тактового Генератора при стабильной системной частоте.

# Внутренний Генератор на 128 кГц

Внутренний Генератор на 128 кГц является маломощным устройством, обеспечивающим тактовую частоту равную 128 кГц. Эта частота является номинальной при Vcc 3V и темпераure 25*C. Этот Генератор может быть выбран в качестве системного программированием CKSEL-битов-перемычек таким образом:"0110-0111". Если в качестве системного Генератора выбран этот, то его время включения определяется установкой SUT-битов-перемычек, как показано в Таблице 11:

**Таблица 11. Время Запуска для Внутреннего Генератора на 128кГц**

| SUT1..0 | Выход из режимов холостого PowerSave(idle) и экономичного PowerDown | Дополнительная задержка после Сброса | Рекомендовано использовать при/с: |
|---|---|---|---|
| 00 | 6СК (тактов) | 14СК | БОД-схема |
| 01 | 6СК | 14СК+4ms | Быстро нарастающее Напряжение |
| 10 | 6СК | 14СК+4ms | Медленно нарастающее напряжение |
| 11 | зарезервировано | | |

# Масштабирующий Регистр-делитель CLKPR

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | CLKPCE | - | - | - | CLKPS3 | CLKPS2 | CLKPS1 | CLKPS0 | CLKPR |
| Чтение/ Запись | Ч/З | Ч | Ч | Ч | Ч/З | Ч/З | Ч/З | Ч/З | |
| Начальные Значения | 0 | 0 | 0 | 0 | Смотри описание Битов | Смотри описание Битов | Смотри описание Битов | Смотри описание Битов | |

**• Бит 7 - CLKPCE: Деление частоты - разрешено**

Бит CLKPCE должен быть записан логической единицей для разрешения изменения CLKPS битов. Бит CLKPCE только обновляется, когда другие биты в этом регистре CLKPR одновременно записываются в НОЛЬ. CLKPCE-бит очищается аппаратно после ЧЕТЫРЕХ тактов процессора или после установки бита CLKPS. Перезапись CLKPS бита в пределах периода ЧЕТЫРЕХ тактов не расширяет этот период аппаратного сброса, бит все равно будет сброшен после того, как он был записан в первый раз.

**• Биты 3..0 - CLKPS3..0: Биты для выбора коэффициента деления тактовой частоты**

Эти биты определяют коэффициент деления между выбранной системной частотой и внутренним системным источником тактирования. Этот бит может быть изменен во время работы для того, чтобы настроить частоту, требуемую приложением. При использовании делителя частоты уменьшается частота синхронизации с периферией. Коэффициенты деления показаны в Таблице 12.

Для того, чтобы не произошло неумышленных изменений частоты, нужно написать процедуру, которая должна следить за изменениями битов CLKPS:

1. Для начала запишите в бит CLKPCE единицу, а все остальные биты в регистре CLKPR сбросьте в ноль.
2. В течении четырех тактов процессора надо записать нужное значение битов CLKPS, пока не произошло аппаратного сброса CLKPCE.

Прерывания во время настройки этого генератора должны быть запрещены, убедитесь, что процедура настройки генератора не была прервана.

Бит-перемычка-CKDIV8 определяет начальное значение для CLKPS битов. Если перемычка CKDIV8 не запрограммирована, тогда CLKPS биты будут сброшены в "0000" нули. Если перемычка CKDIV8 запрограммирована, тогда CLKPS биты будут сброшены в "0011" значения, определяя этим коэффициент деления равный 8 при старте контроллера. Эта особенность должна быть использована в случае, когда выбранный источник тактирования имеет более высокую частоту, чем требуемая максимальная частота устройства для данного режима.

Обратите внимание, что можно записать любое значение в биты CLKPS независимо от установок бита перемычки-CKDIV8. Необходимо удостовериться, что коэффициент деления частоты был выбран достаточным, если источник тактирования имеет более высокую частоту, чем надо. Контроллер изначально выпускается с запрограммированным битом-перемычкой CKDIV8.

**Таблица 12. Выбор делителя частоты**

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

# Управление потреблением мощности и РежимыСна

Режим сна позволяет отключить в приложении неиспользуемые модули микроконтроллера, таким образом сохраняя энергию. Семейство AVR обеспечивает разные режимы сна, позволяющие пользователю адаптировать потребление мощности к требованиям приложения.

Для включения любого из трех режимов сна надо записать единицу в бит SE в регистре SMCR и затем выполнить инструкцию SLEEP. Биты SM0 и SM1 в регистре MCUCR определяют, какой из режимов будет активирован инструкцией SLEEP (режимы: Idle-холостой, Power-down-экономичный, или Standby-сна). Смотрите Таблицу 13. Если в то время, пока контроллер спит, происходит РАЗРЕШЕННОЕ прерывание, тогда контроллер выходит из режима сна и выполняет какую-то работу.

Когда контроллер просыпается, он приостанавливается на четыре такта, время которых добавляется к времени его загрузки, выполняется вызов подпрограммы обрабатывающей прерывание и после этого продолжается выполнение инструкций, следующих за инструкцией SLEEP, собственно, которая и ввела контроллер в режим сна. Содержание Регистрового Файла и памяти SRAM остается нетронутым, когда устройство выходит из режима сна. Если произойдет Сброс во время сна, после этого контроллер стартует и выполняет программу с Вектора Сброса.

Рисунок 11 на странице 21 показывает различные тактируемые модули, а,также, их расположение. Этот рисунок может оказаться полезным при выборе нужного режима сна.

# MCUCR – Регистр управления МК (микроконтроллером)

Регистр Управления Режимом Сна содержит управляющие биты для регулирования потребляемой мощности.

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | PUD | SM1 | SE | SM0 | ISC11 | ISC10 | ISC01 | ISC00 | MCUCR |
| Чтение/ Запись | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | |
| Начальные Значения | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**• Биты 6, 4 - SM1..0: Режимы Сна**

Эти биты выбирают один из четырех доступных РежимовСна, как показано в Таблице 13.

**Таблица 13 – Режимы Сна**

| SM1 | SM0 | Режимы Сна |
|---|---|---|
| 0 | 0 | Холостой-Idle |
| 0 | 1 | Экономичный-PwerDown |
| 1 | 1 | Экономичный-PowerDown |
| 1 | 0 | Режим Сна-Standby |

**Примечание 1: Режим Сна (именно Standby) рекомендуется использовать только при использовании внешнего резонатора для Генератора.**

# ХолостойРежим-Idle

Когда биты SM1..0 записаны в нули "00", инструкция SLEEP вводит МК в ХолостойРежим Сна, при этом останавливается ЦПУ, но при этом разрешена и продолжается работа таких модулей, как: UART, Аналоговый Компаратор, АЦП, USI-интерфейс, Таймеры/Счетчики, СторожевойТаймер и СистемаПрерываний.

**ХолостойРежим** позволяет МК пробуждаться от внешних прерываний, это происходит так же, как пробуждение от внутренних прерываний, к примеру при Переполнении Таймера/Счетчика или при окончании передачи данных через UART. Если прерывание от АналоговогоКомпаратора не требуется, то его можно отключить установкой ACD-бита в Регистре Управления Аналоговым Компаратором - ACSR. Это уменьшит потребляемую мощность в ХолостомРежиме.

# Режим СохраненияЭнергии (Экономичный)

Когда биты SM1..0 записаны в 01 или 11, инструкция SLEEP переводит МК в режим Сохранения нергии. В этом режиме внешний генратор останавливается, в то время, как внешние прерывания, такие, как Определение Начала Передачи Данных по интерфейсу USI и СторожевойТаймер, продолжают работать (если, конечно ОНИ разрешены). Вывести МК из этого режима могут только следующие прерывания: ВнешнийСброс, Сброс от СторожевогоТаймера, Сброс BOD-схемой, Прерывание от USI-интерфейса при определении условия начала передачи данных, изменением внешнего уровня сигнала на выводе INT0 или прерывание на другом выводе, служащим источником прерывания, определенном пользователем. Этот РежимСна основан на том, что происходит остановка всех тактовых генераторов, позволяя работать только модулям не требующим тактовой частоты.

Обратите внимание, что, если в качестве ВнешнегоПрерывания для вывода МК из РежимаСохраненияЭнергии-PowerDown, используется изменение уровня сигнала на каком-то из выводов то заданный уровень сигнала должен присутствовать некоторое минимальное время. Обратитесь к разделу "Внешние Прерывания" на стр. 58 за уточнением этой информации.

При выходе из РежимаСохраненияЭнергии происходит еще нкоторая временная задержка до того, как МК полностью проснется. Это позволяет ТактовомуГенератору перезапуститься и стабилизировать свою работу после того, как он был остановлен. Период пробуждения МК определяется теми же самыми битами-Перемычками, которые определяют Время Дополнительной Задержки ПослеСброса, это описано ранее в раздееле "Источник Тактирования" на стр. 22 и в табл.5 на стр.24.

РежимСна-StandBy

Если биты SM1..0 имеют значения "10" и выбран внешний тактовый генератор на кристалле/резонаторе, то инструкция SLEEP переводит МК в РежимСна (StandBy). Этот режим идентичен режиму СохраненияЭнергии-PowerDown (или еще по-другому - ЭкономииЭнергии) за тем исключением, что в этом режиме-StandBy ТактовыйГенератор остается в рабочем состоянии. Из этого РежимаСна МК выходит за ШЕСТЬ тактовых циклов.

**Таблица 14 – Активные Области тактирования и Источники Пробуждения**

| | Активные Области Тактирования | | | Генераторы | Источники пробуждения | | | |
|---|---|---|---|---|---|---|---|---|
| Режимы Сна | ТактыCPU | ТактыFLASH | ТактыI/O | Разрешено | INT0,INT1 или PCINT0..PCINT7 | Условие старта USI | SPM/EEPROM | Другие I/O |
| Холостой – Idle | | | X | X | X | X | X | X |
| Экономии-PowerDown | | | | | X(2) | X | | |
| Сна – StandBy (1) | | | | X | X(2) | X | | |

**Примечания: 1. Рекомендовано при использовании внешнего кристалла или резонатора.**
**2. Для INT0 - только прерывание по уровню.**

# Экономия Потребляемой Энергии

Приходится рассматривать несколько проблем одновременно при попытке минимизировать мощность потребления AVR -контроллерами. В общем, все режимы сна должны использоваться максимально эффективно, и тогда, когда это только возможно и режим сна надо выбирать таким, чтобы обеспечить работу только нужных модулей устройства. Все ненужные модули должны быть отключены. В частности, следующие модули могут нуждаться в особом рассмотрении при попытке достигнуть низшего уровня потребления энергии.

# Аналоговый Компаратор (АК)

При входе МК в ХолостойРежим-Idle, АналоговыйКомпаратор должен быть отключен, если он не используется. В других режимах сна АК отключается автоматически. Однако, если АК настроен на использование Внутреннего Источника Опорного Напряжения в качестве входного, тогда АК будет сам отключаться во всех режимах сна. В других случаях Внутренний Источник Опорного Напряжения будет активен независимо от какого-либо режима сна. Обратитесь к разделу "Аналоговый Компаратор" на странице 148 за более детальной информацие об АК.

# BOD-детектор (Brown-out Detector)

(BOD-детектор или схема защиты (слежения) от пониженного напряжения питания)

Если BOD-детектор не нужен в приложении то этот модуль необходимо отключить. Если BOD-детектор включен перемычками-BODLEVEL, тогда он будет работать во всех режимах сна и, следовательно, всегда потреблять энергию. Обратитесь к разделу "BOD-детектор" на странице 34 для более детальной информации.

# Внутренний Источник Опорного Напряжения (ВИОН)

Внутренний Источник Опорного Напряжения должен быть разрешен, когда используется BOD-детектор или Аналоговый Компаратор. Если эти модули отключены, как рекомендовано в предыдущих разделах,
ВИОН должен быть тоже отключен и тогда он не будет потреблять энергию. Когда ВИОН снова надо включить, пользователь должен сначала позволить ему нормально запуститься, прежде чем ВИОН можно будет использовать. Если ВИОН не выключался на период сна МК то ВИОН может быть использован без промедления. Обратитесь к разделу "Внутренний Источник Опорного Напряжения" на странице 37 за более детальной информацией.

# Сторожевой Таймер (СТ)

Если СторожевойТаймер не используется в приложении, тогда этот модуль лучше отключить. Если этого не сделать, ОН будет работать в любом режиме сна и, следовательно, будет всегда потреблять энергию. В самом экономичном режиме сна СТ внесет значительный вклад в потреблении тока. Обратитесь к разделу "Прерывания" на странице 41 за более детальной информацией о том, как настраивать СторожевойТаймер.

# Выводы Портов вода/вывода

При входе в различные режимы сна, все выводы портов в/в будут сконфигурированы на минимальное потребление энергии. Очень важно перед этим убедиться, что никакие выводы не должны управлять резистивными нагрузками. В режиме сна, когда отключается ТактовыйГенератор, все буферы портов в/в тоже отключаются. Это служит гарантией того, что порты в/в не потребляют тока, когда это не требуется. В некоторых случаях, при выходе из различных режимов сна, порты в/в нуждаются в настройке и только после этого ОНИ будут работать исправно. Обратитесь к разделу "Разрешение Цифрового Ввода и Режимы Сна" на странице 49 за более детальной информацией о том, к каким выводам относится вышесказанное. Необходимо помнить, что, если входные буферы портов в/в включены и входной сигал на каком-либо выводе изменяется или является аналоговым сигналом достигающим уровня Vcc/2 то входные буферы будут потреблять чрезмерную мощность.

Для выводов, настроенных на работу с аналоговыми сигналами, цифровой входной буфер отключатся. При аналоговых сигналах, достигающих уровня Vcc/2 на выводе порта в/в, возможно значительное потребление тока, даже в активном режиме вывода. Цифровой входной буфер может быть отключен соответствующими битами, записываемыми в регистр DIDR "Digital Input Disable Register - DIDR", о чем рассказано на странице 149.

# Контроль за Системой и Перезагрузка

После перезагрузки все регистры в/в устанавливаются в их начальное значение, затем, старт и выполнение программы с ВектораСброса ResetVector. Инструкция, расположенная возле ВектораСброса, может совершить безусловный переход (RJMP) на подпрограмму, обрабатывающую прерывание по Сбросу. Если в программе не предусмотрено использование прерываний, тогда ВекторПрерывания не используется и код бесконечного программного цикла располагается на месте ВектораПрерывания. На Рисунке14 показана Логическая Схема Сброса. В Таблице15 определены электрические параметры Схемы Сброса.

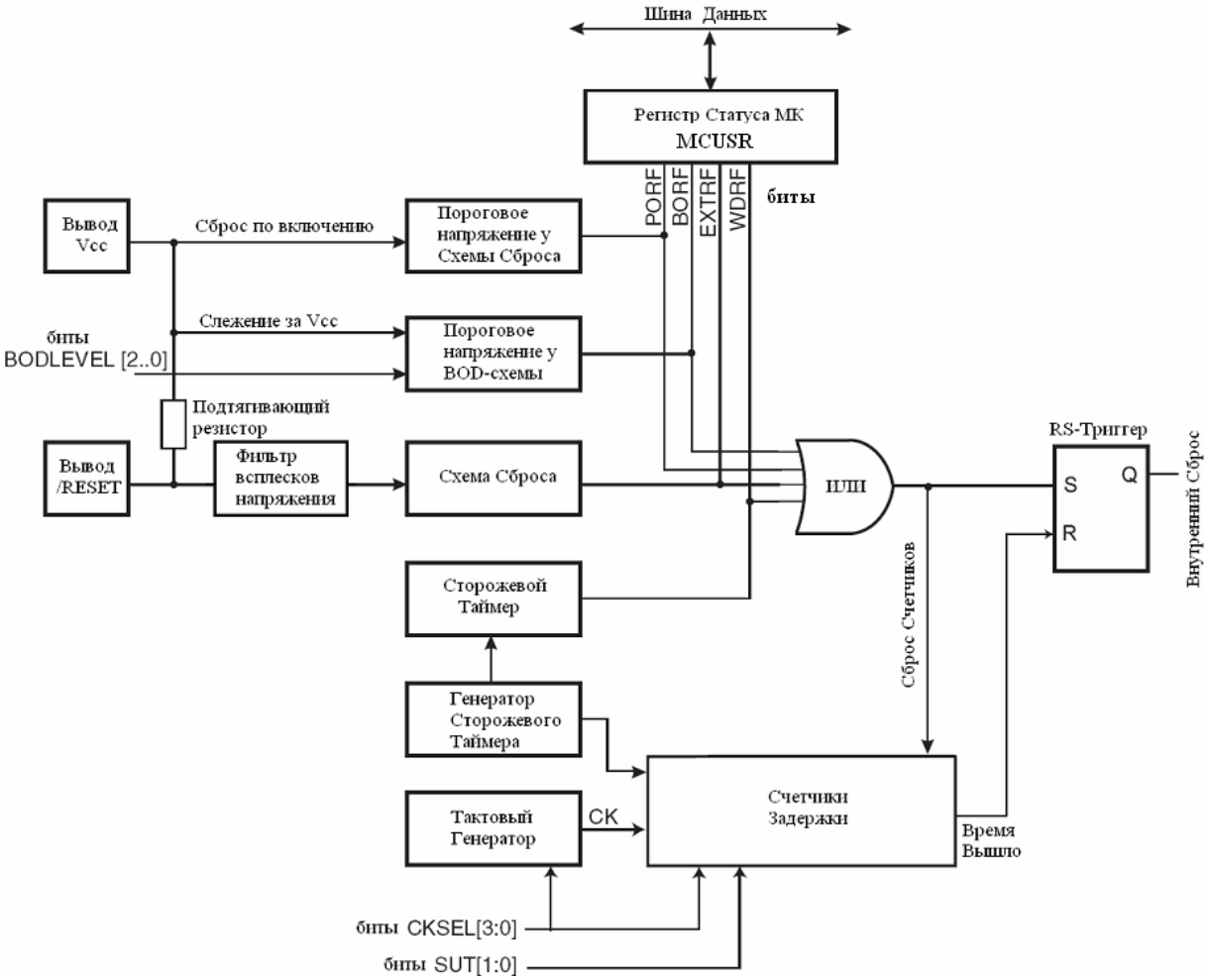



Рисунок 14 Логическая Схема Сброса МК

Порты в/в МК немедленно сбрасываются в их значения по-умолчанию, когда активируется Схема Сброса. Она не требует тактовой синхронизации для запуска. После того, как Схема Сброса выключается, включается Счетчик Задержки, призванный отдалить время Внутренней Перезагрузки. Это позволяет Тактовому Генератору стабилизировать свою работу перед тем, как начать выполнение программы. Период задержки у Счетчика Задержки определяется пользователем через установку битов-перемычек SUT и CKSEL. Выбор периода задержки описан на странице 22.

**Источники Сброса:**

ATtiny2313 имеет 4 источника сброса

- **Сброс по включению**. МК постоянно перезагружается, когда уровень Vcc у источника напряжения ниже порогового напряжения $V_{POT}$, разрешающего включение МК.
- **Внешний Сброс**. МК перезагружается, когда на выводе /RESET низкий уровень напряжения присутствует дольше, чем определенный минимум длительности импульса.
- **Сброс от Сторожевого Таймера**. МК перезагружается, когда период у СТ истекает, при этом работа Стдолжна быть разрешена, а прерывания от СТ должны быть запрещены.
- **Сброс от BOD-схемы**. МК перезагружается, когда значение Vcc ниже установленного порогового напряжения $V_{BOT}$, при этом работа BOD-детектора должна быть разрешена.

**Таблица 15 – Условия Сброса МК**

| Переменная | Параметры | Условия работы | Минимальное значение (1) | Нормальное Значение (1) | Максимальное Значение (1) | Единицы измерения |
|---|---|---|---|---|---|---|
| $V_{POT}$ | Граница напряжения для Сброса по Включению (нарастающее) | $T_A$=-40 – 58*C | | 1,2 | | Вольт |
| | Граница напряжения для Сброса по Включению (спадающее)(2) | $T_A$=-40 – 58*C | | 1,1 | | Вольт |
| $V_{RST}$ | Граница значения напряжения на выводе Сброса | Vcc=1,8 – 5,5 | 0,1 Vcc | | 0,9 Vcc | Вольт |
| $t_{RST}$ | Минимальная длительность импульса низкого уровня на /RESET | Vcc=1,8 – 5,5 | | | 2,5 | мкСек |

Примечание:
1. Эти значения являются только рекомендованными/предварительными. Действительные значения представлены в TBD.
2. Схема Сброса по Включению не будет работать, если значение источника напряжения будет ниже $V_{POT}$ (в режиме спадающего напряжения).

# Схема Сброса по Включению - "Power-on Reset" (POR-схема)

POR-схема, вырабатывающая импульсы Сброса, расположена на кристалле МК. Уровни напряжения, при которых она работает, описаны в Таблице 15. POR-схема активируется всякий раз, когда значения источника Vcc ниже положенного уровня. POR-схема может быть использована как для сброса МК, так и для контроля источника Vcc. POR-схема гарантирует сброс системы. После достижения порога нормального напряжения, установленного POR-схемой, включается Счетчик Задержки, который определяет, как долго МК будет находиться в состоянии Сброса после того, как Vcc достигло приемлемого уровня. RESET-сигнал будет вырабатываться каждый раз, без всякой задержки, когда уровень Vcc упадет ниже определенного уровня.

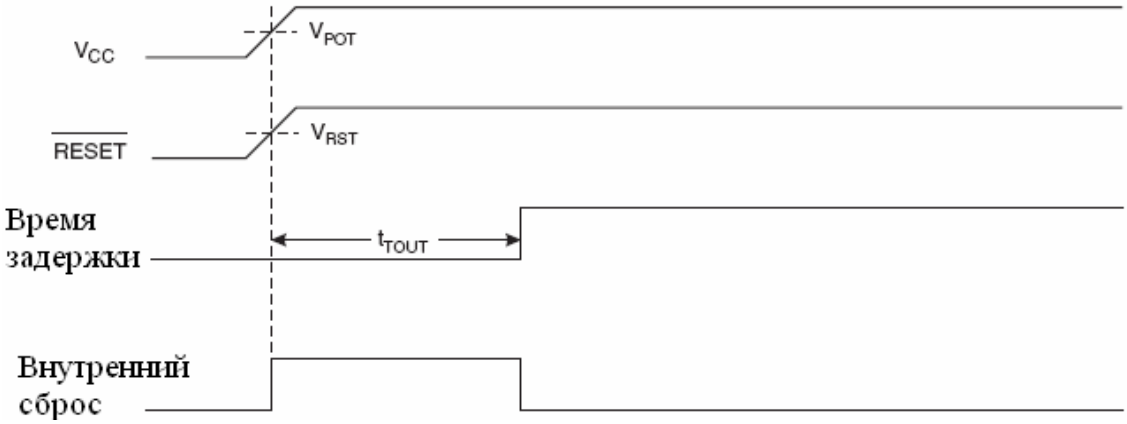



Рисунок 15 – Загрузка МК, RESET привязан к Vcc

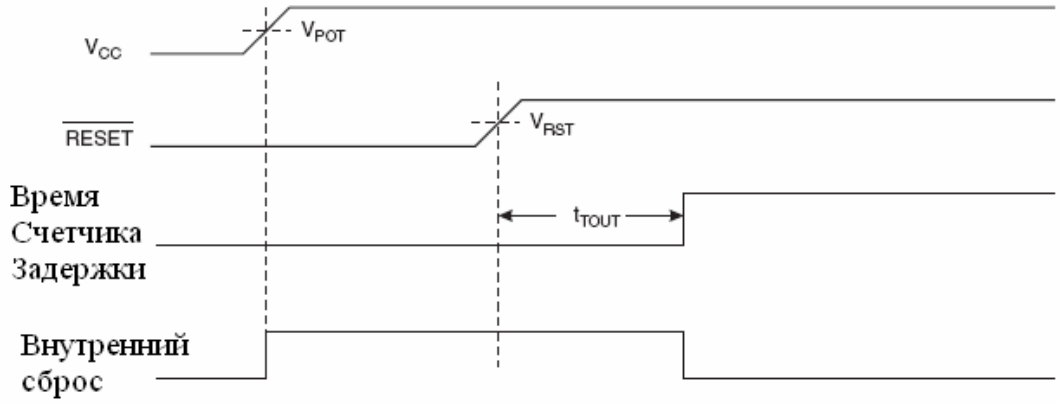



Рисунок 16 – Временная диаграмма Периода Включения МК

# Внешний Сброс (ВС)

ВС возникает при наличии логического нуля на выводе RESET. Если на этом выводе присутствует низкоуровневый сигнал дольше, чем положено (смотри Таблицу 15), будет сгенерирован Сброс МК, даже, если Тактовый Генератор не запущен. Сброс не гарантирован, если импульс высокого уровня отсутствует меньше, чем установлено в Таблице15. Когда сигнал нарастает и достигает номинала $V_{RST}$, тогда Счетчик Задержки, по истечению своего периода работы $t_{TOUT}$, позволит включить МК.

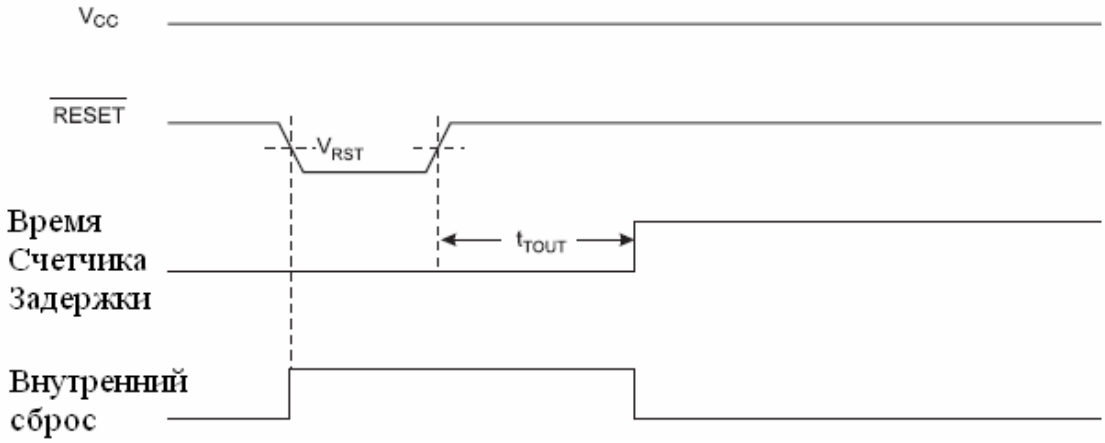



Рисунок 17 – Внешний Сброс во время Работы МК

# BOD-детектор (Схема Слежения за Питанием)

ATtiny2313 имеет встроенный BOD-детектор, это схема, которая следит за состоянием уровня Vcc в реальном времени работы МК, постоянно сравнивая текущее значение Vcc с установленным фиксированным значением. Уровень установленного фиксированного значения сравнения определяется установкой битов-перемычек "BODLEVEL". Уровень фиксированного значения сравнения имеет свой гистерезис, что позволяет работать BOD-схеме в целом диапазоне значений, гарантируя отсутствие реакции схемы на всплески напряжения. Гистерезис рассчитывается так: $V_{BOT+} = V_{BOT} + V_{HYST}/2$ и $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

**Таблица 16 – Установка BODLEVEL-перемычек (1)**

| Биты-перемычки BODLEVEL 2..0 | Минимальное напряжение $V_{BOT}$ | Нормальное напряжение $V_{BOT}$ | Максимальное напряжение $V_{BOT}$ | Единицы измерения |
|---|---|---|---|---|
| 111 | BOD-схема отключена | | | Вольт |
| 110 | | 1,8 | | Вольт |
| 101 | | 2,7 | | Вольт |
| 100 | | 4,3 | | Вольт |
| 011 | Зарезервировано | | | |
| 010 | | | | |
| 001 | | | | |
| 000 | | | | |

Примечание:

1. Значение $V_{BOT}$ может быть ниже определенного минимума для некоторых устройств. МК, те, у которых актуален этот случай, тестируются во время производства. Это гарантирует, что Сброс BOD-схемой выполняется до того, как Vcc упадет до уровня напряжения, при котором корректное выполнение контроллером задач не гарантируется. Тестирование производится с использованием уровней, устанавливаемых через биты-перемычки BODLEVEL.

**Таблица 17 – Характеристики BOD-детектора**

| Переменная | Параметры | Минимал. | Нормал. | Максимал. | Единици измерения |
|---|---|---|---|---|---|
| $V_{HYST}$ | Гистерезис BOD-схемы | - | 50 | - | мВ |
| $t_{BOD}$ | Минимальная длительность импульса, на который реагирует BOD-схема. | - | 2 | - | нСек |

Когда работа BOD-схемы разрешена и Vcc уменьшается до значения ниже порога $V_{BOTIN}$ -Рисунок 18- , немедленно генерируется сброс от BOD-схемы . Когда Vcc увеличивается до уровня выше порога $V_{BOT+}$ -Рисунок 18- , Счетчик Задержки разрешит МК включиться по истечению периода положенной задержки $t_{TOUT}$ .BOD-схема определит напряжение Vcc как не удовлетворительное, если уровень Vcc будет ниже положенного уровня в течении периода времени большего, чем $t_{BOD}$, как показано в Таблице 15.

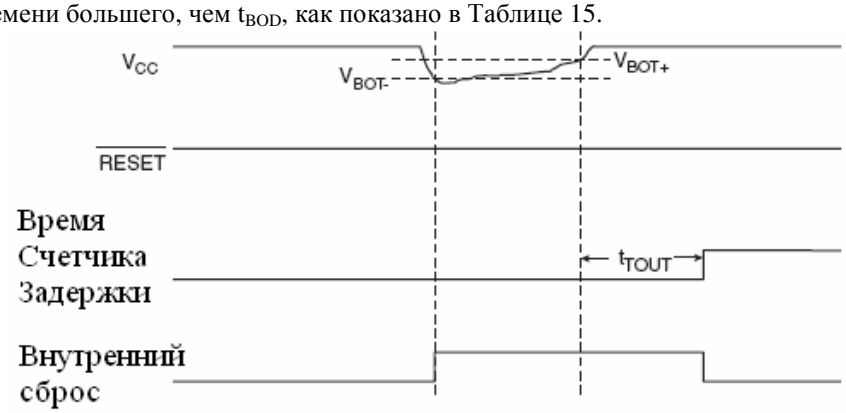



Рисунок 18 – Сброс BOD-схемой во время работы МК

# Сброс от Сторожевого Таймера «WDT»

Когда время Сторожевого Таймера истекает, он генерирует короткий импульс Сброса, длительностью в один период Тактовой Частоты Системного Генератора. По спадающему фронту этого импульса Сброса начинает свою работу Таймер Задержки, который считает время $t_{TOUT}$ . За более подробной информаций о работе Сторожевого Таймера обращайтесь к странице 43.

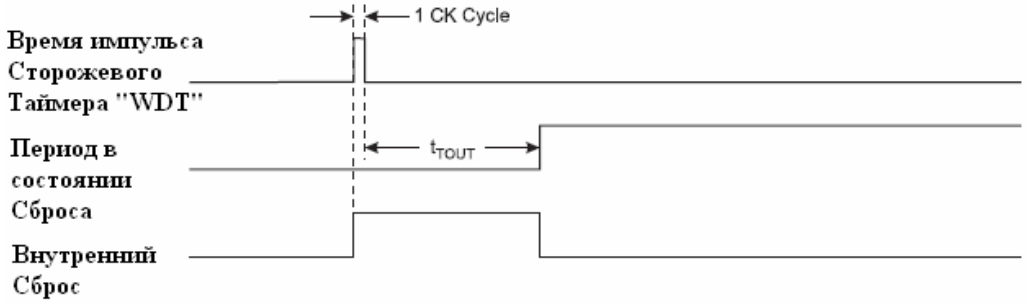



Рисунок 19 – Сброс от Сторожевого Таймера во время работы МК

# Регистр Статуса МК "MCUSR"

Регистр Статуса МК предоставляет информацию о том, кто выполнил Сброс.

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | WDRF | BORF | EXTRF | PORF | MCUSR |
| Чтение/ Запись | Ч | Ч | Ч | Ч | Ч/З | Ч/З | Ч/З | Ч/З | |
| Начальные значения | 0 | 0 | 0 | Смотри описание битов | Смотри описание битов | Смотри описание битов | Смотри описание битов | Смотри описание битов | |

**• Bit 3 - WDRF: Watchdog Reset Flag: Флаг сброса от Сторожевого Таймера**

Бит устанавливается в 1 после сброса от WDT. Бит устанавливается в 0 после выключения питания МК или записью в него логического нуля.

**• Bit 2 - BORF: Brown-out Reset Flag: Флаг Сброса от BOD-схемы**

Бит устанавливается в 1 после сброса от BOD-схемы . Бит устанавливается в 0 после выключения питания МК или записью в него логического нуля.

**• Bit 1 - EXTRF: External Reset Flag: Флаг сброса от Внешнего Источника**

Бит устанавливается в 1 после сброса от Внешнего Источника. Бит устанавливается в 0 после выключения питания МК или записью в него логического нуля.

**• Bit 0 - PORF: Power-on Reset Flag: Флаг Сброса по Включению Питания**

Бит устанавливается в 1 после сброса по Включению Питания. Бит устанавливается в ноль ТОЛЬКО записью в него логического нуля.

Чтобы использовать RESET-флаги для идентификации источника Сброса, пользователь должен прочитать, а затем очистить MCUCR-регистр и сделать это надо настолько быстро, как только возможно. Если MCUCR-регистр успевает очищаться до выполнения каких-либо Сбросов то Источник Сброса может быть пойман/вычислен по установленному флагу.

# Встроенный Источник Опорного Напряжения (ИОН)

ATtiny2313 имеет встроенный ИОН. ИОН активно используется BOD-схемой и может быть использован в качестве сигнала, подаваемого на вход Аналогового Компаратора.

## Сигнал Включения ИОНа и Время Включения ИОНа

ИОН имеет некоторое время запуска, которое может повлиять на приложения, где он будет использоваться. Время запуска приведено в Таблице 18. В целях энергосбережения, ИОН не всегда находится во включенном состоянии.

ИОН включается в следующих ситуациях:

1. Когда разрешена работа BOD-схемы (путем программирования BODLEVEL-перемычек [2..0])
2. Когда ИОН подключен ко входу Аналогового Компаратора (установкой битов ACBG в Регистре ACSR).

Таким образом, когда BOD-схема отключена, после установки бита ACBG пользователь должен всегда позволить ИОНу нормально запуститься, прежде, чем использовать Аналоговый Компаратор. Для снижения энергопотребления в Экономичном Режиме Сна -PowerDown- пользователь может исключить ДВА режима работы, описанных выше, чтобы гарантировать, что ИОН будет выключен до входа в PowerDown-режим.

**Таблица 18 – Характеристики Встроенного ИОНа (1)**

| Переменная | Параметры | Условия работы | Минимальное значение (1) | Нормальное Значение (1) | Максимальное Значение (1) | Единицы измерения |
|---|---|---|---|---|---|---|
| $V_{BG}$ | Диапазон изменения Опорного Напряжения | Vcc=2.7 В 25*C | 1 | 1.1 | 1.2 | Вольт |
| $t_{BG}$ | Время Запуска ИОНа | Vcc=2.7 В 25*C | - | 40 | 70 | мкСек |
| $I_{BG}$ | Ток, потребляемый ИОНом | Vcc=2.7 В 25*C | - | 15 | - | мкА |

Примечание:

1. Эти значения являются только рекомендованными/предварительными. Действительные значения представлены в TBD.

# Сторожевой Таймер “WDT”

ATtiny2313 имеет на борту Усовершенствованный Сторожевой Таймер (WDT).

Его главными особенностями являются:

1. Тактирование от отдельного Встроенного на кристалл Генератора.
2. Три режима работы:
2.1 Режим Прерывания
2.2. Режим Сброса Системы
2.3 Режим Прерывания и Сброса Системы
3. Выбираемый Период Отсчета от 16 мСек до 8 Сек
4 Возможность аппаратного включения WDT навсегда – для надежности.

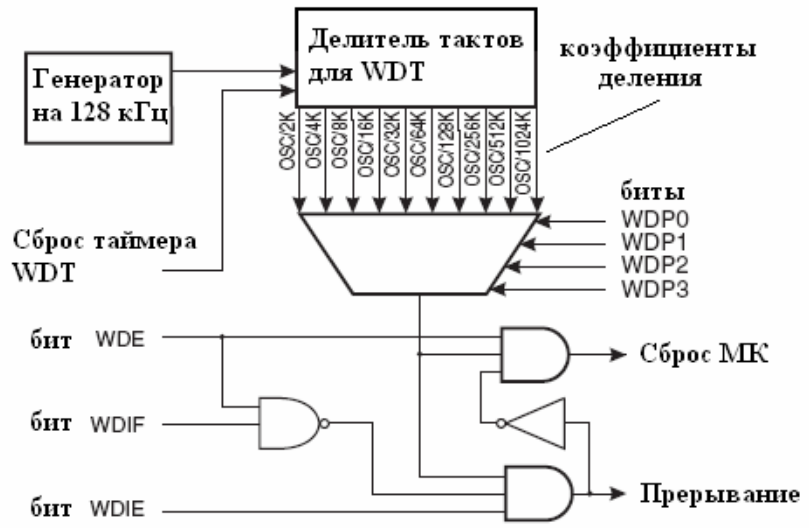



Рисунок 20 – Структура Сторожевого Таймера WDT

Сторожевой Таймер -WDT- , это таймер, считающий такты от отдельного встроенного на кристалл генератора на 128 кГц. Сторожевой Таймер генерирует Прерывание или Системный Сброс, когда счетчик достигает заданного значения. В нормальном режиме система, использующая WDT, должна обеспечивать сброс С.Таймера до того, как его счетчик досчитает до заданного значения. Если система не сбросит С.Таймер, он сгенерирует Прерывание или Сброс Системы.

**В Режиме Прерывания** WDT генерирует сигнал прерывания, когда счетчик заполняется полностью. Такое прерывание может быть использовано в качестве пробуждающего сигнала для выхода из режимов сна или в качестве обычного системного таймера. С.Таймер можно использовать для ограничения времени работы контроллера, генерируя сигнал прерывания, если контроллер долго не отвечает и не сбрасывает счетчик WDT.

**В Режиме Сброса Системы** WDT генерирует сигнал Сброса Системы, когда счетчик заполняется полностью. Обычно этот режим используют для предотвращения зависания системы, в случае выхода кода из под контроля.

**Третий режим, это Режим Прерывания и Сброса Системы**, который комбинирует два других режима, первый - генерирует Прерывание, а затем переключается в Режим Сброса Системы. Этот Третий режим может обеспечить безопасную перезагрузку, позволив сохранить критически важные данные перед тем, как перезагрузить систему.

Сторожевой Таймер может быть включен навсегда, прошиванием перемычки WDTON, в этом случае WDT будет работать в режиме Сброса Системы. При прошивании «плавкой» перемычки WDTON, автоматически выбирается режим Системного Сброса, соответствующий бит WDE получает значение 1 и блокируется, бит Режима Прерывания WDIE получает значение 0 и тоже блокируется.

Далее, чтобы гарантировать безопасное выполнение программы, изменения в настройках WDT должны быть выполнены в определенной последовательности.

Последовательные шаги очищения бита WDE и смены периода подсчета WDTтакие:

1. В том же режиме запишите логическую единицу в биты WDCE и WDE. Логическая единица должна быть записана в бит WDE независимо от предыдущего значения этого бита.

2. В течении следующих четырех тактов Системного Генератора запишите бит WDE единицей и выберите записью единицы желаемый бит-делитель WDP, но при этом бит WDCE должен быть очищен. Это должно быть сделано в одном режиме.

Следующий пример кода показывает как на Ассемблере и С-функции включать WDT. Эти примеры предполагают, что прерывания отключены глобально, так что прерывания не произойдут в течении выполнения этих функций.

Assembly Code Example[1]

```
WDT_off:
  ; Turn off global interrupt
  cli
  ; Reset Watchdog Timer
  wdr
  ; Clear WDRF in MCUSR
  in    r16, MCUSR
  andi  r16, (0xff & (0<<WDRF))
  out   MCUSR, r16
  ; Write logical one to WDCE and WDE
  ; Keep old prescaler setting to prevent unintentional time-out
  in    r16, WDTCSR
  ori   r16, (1<<WDCE) | (1<<WDE)
  out   WDTCSR, r16
  ; Turn off WDT
  ldi   r16, (0<<WDE)
  out   WDTCSR, r16
  ; Turn on global interrupt
  sei
  ret
```

C Code Example[1]

```
void WDT_off(void)
{
  __disable_interrupt();
  __watchdog_reset();
  /* Clear WDRF in MCUSR */
  MCUSR &= ~(1<<WDRF);
  /* Write logical one to WDCE and WDE */
  /* Keep old prescaler setting to prevent unintentional time-out
*/
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  /* Turn off WDT */
  WDTCSR = 0x00;
  __enable_interrupt();
}
```

**Примечание:** 1. Кодовые примеры предполагают, что нужные заголовочные файлы уже включены в программу. **Примечание:** Если работа WDT была разрешена случайно, тогда устройство будет перезагружено, после чего WDT останется во включенном состоянии. Если код программы не предусматривает подпрограмму

обработки WDT , это может привести к зацикливанию на внутреннем сбросе МК от WDT по заполнению счетчика WDT. Чтобы избежать эту ситуацию, программный код должен постоянно очищать флаг WDRF и бит управления WDE в Подпрограмме Инициализации (начальной настройке МК при включении), даже, если WDT не используется. Следующий пример кода на Ассемблере и языке С показывает, как изменять время заполнения таймера WDT.

Assembly Code Example[1]

```
WDT_Prescaler_Change:
  ; Turn off global interrupt
  cli
  ; Reset Watchdog Timer
  wdr
  ; Start timed sequence
  in    r16, WDTCSR
  ori   r16, (1<<WDCE) | (1<<WDE)
  out   WDTCSR, r16
  ; --  Got four cycles to set the new values from here -
  ; Set new prescaler(time-out) value = 64K cycles (~0.5 s)
  ldi   r16, (1<<WDE) | (1<<WDP2) | (1<<WDP0)
  out   WDTCSR, r16
  ; --  Finished setting new values, used 2 cycles -
  ; Turn on global interrupt
  sei
  ret
```

C Code Example[1]

```
void WDT_Prescaler_Change(void)
{
  __disable_interrupt();
  __watchdog_reset();
  /* Start timed  equence */
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  /* Set new prescaler(time-out) value = 64K cycles (~0.5 s) */
  WDTCSR  = (1<<WDE) | (1<<WDP2) | (1<<WDP0);
  __enable_interrupt();
}
```

Примечание: 1. Пример кода предполагает, что соответствующие заголовочные файлы включены в проект.

Примечание: WDT должен быть сброшен перед тем, как пользователь будет менять WDP бит, так как смена этого бита без сброса WDT может привести к более раннему заполнению таймера WDT, чем планировалось.

# Регистр Управления Сторожевым Таймером (WDTCSR)

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | WDIF | WDIE | WDP3 | WDCE | WDE | WDP2 | WDP1 | WDP0 | WDTCSR |
| Чтение/ Запись | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | |
| Начальные значения | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | |

**• Bit 7 - WDIF: Флаг Прерывания от WDT**

Этот бит устанавливается в единицу, когда происходит переполнение таймера WDT, при условии, что WDT настроен на работу в Режиме Прерывания. Бит WDIF очищается аппаратно, когда выполняется переход на соответствующий вектор прерывания. Также, этот бит можно очистить, записав в него логическую единицу. Если бит I в регистре SREG установлен, то есть, разрешены прерывания, тогда установка бита WDIE вызывает прерывание от WDT.

**• Bit 6 - WDIE: разрешение Прерывания от WDT**

Если записать в этот бит логическую единицу, то при условии, что разрешены прерывания, будет разрешено генерировать прерывание от WDT. Теперь, в дополнение к этим установкам, если еще очистить бит WDE в ноль, тогда Сторожевой Таймер будет работать в Режиме Прерывания, и будет вызываться соответствующее прерывание при переполнении таймера.

Если бит WDE установлен в 1, тогда WDT работает в Третьем Режиме, совмещая Режим Прерывания и Режим Сброса Системы. Первое переполнение таймера WDT приводит к установке бита WDIF. Затем, при переходе на соответствующий вектор прерывания, биты WDIE и WDIF автоматически аппаратно очищаются (WDT переключается в Режим Сброса Системы). Сброс этих битов полезен, так как позволяет защититься от действий WDT во время обработки вызванного прерывания. Для того, чтобы WDT и далее оставался в Третьем режиме, бит WDIE должен устанавливаться после каждого прерывания. Однако, его установка не должна выполняться в подпрограмме, обрабатывающей прерывание, т.к. это может поставить под угрозу систему в целом. Ведь, если прерывание не было вызвано до следующего переполнения таймера WDT, то бит WDIE не установился и произойдет Сброс Системы.

**Таблица 19 – Настройка Сторожевого Таймера**

| WDTON | WDE | WDIE | Режим | Действие при переполнении таймера WDT |
|---|---|---|---|---|
| 0 | 0 | 0 | Остановлен | Нет |
| 0 | 0 | 1 | Режим Прерывания | Вызов Прерывания |
| 0 | 1 | 0 | Режим Сброса Системы | Сброс |
| 0 | 1 | 1 | Третий режим – Режим Прерывания и Режим Сброса Системы | Сначала Прерывание, а затем переход в Режим Сброса Системы |
| 1 | X | X | Режим Сброса Системы | Сброс Системы |

**• Bit 4 - WDCE: Бит разрешения настройки WDT**

Этот бит используется для изменения бита WDE и выбора бита предделителя. Для того, чтобы очистить бит WDE бит и/или сменить бит предделителя, нужно сначала установить бит WDCE.
Если в WDCE записана лог. единица, то она будет очищена аппаратно, спустя четыре тактовых цикла.

**• Bit 3 - WDE: Разрешить WDT сбрасывать систему**

Бит WDE игнорируется флагом WDRF в регистре MCUCR. Это означает, что бит разрешения прерывания WDE устанавливается всегда, когда устанавливается флаг прерывания WDRF. Для того, чтобы очистить бит WDE, нужно сначала очистить флаг WDRF. Этим гарантируется постоянный сброс системы, пока присутствует условие краха системы и безопасная загрузка системы после исчезновения угрозы.

**• Bit 5, 2..0 - WDP3..0: Биты Предделителя WDT 3, 2, 1 и 0**

Биты WDP3..0 определяют выбор Предделителя (Коэффициента деления), во время работы WDT. Какие бывают Предделители и соответствующие им периоды переполнения приведены в Таблице 20 на странице 42.

**Таблица 20 – Выбор Предделителя (Коэффициента деления) WDT**

| WDP3 | WDP2 | WDP1 | WDP0 | Количество тактов таймера WDT | Типичное время для Vcc=5.0 В |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 2 К (2048) тактов | 16 мСек |
| 0 | 0 | 0 | 1 | 4 К (4096) тактов | 32 мСек |
| 0 | 0 | 1 | 0 | 8 К (8192) тактов | 64 мСек |
| 0 | 0 | 1 | 1 | 16 К (16384) тактов | 0.125 Сек |
| 0 | 1 | 0 | 0 | 32 К (32768) тактов | 0.25 Сек |
| 0 | 1 | 0 | 1 | 64 К (65536) тактов | 0.5 Сек |
| 0 | 1 | 1 | 0 | 128 К (131072) тактов | 1.0 Сек |
| 0 | 1 | 1 | 1 | 256 К (262144) тактов | 2.0 Сек |
| 1 | 0 | 0 | 0 | 512 К (524288) тактов | 4.0 Сек |
| 1 | 0 | 0 | 1 | 1024 К (1048576) тактов | 8.0 Сек |
| 1 | 0 | 1 | 0 | Зарезервировано | |
| 1 | 0 | 1 | 1 | | |
| 1 | 1 | 0 | 0 | | |
| 1 | 1 | 0 | 1 | | |
| 1 | 1 | 1 | 0 | | |
| 1 | 1 | 1 | 1 | | |

**Примечание переводчика: секунда может обозначаться как буква «с» или сокращенно «Сек».**

# Прерывания

Этот раздел описывает специфику обработки прерываний, выполненную в ATtiny2313. За общей информацией о прерываниях в AVR обратитесь к разделу "Сброс и Обработка Прерываний", страница 11.

## Векторы Прерывания в ATtiny2313

**Таблица 21 – Векторы Прерывания**

| Номер Вектора | Адрес в Программе | Источник Gрерывания | Описание Прерывания |
|---|---|---|---|
| 1 | 0x0000 | RESET | Прерывание на внешнем выводе “RESET”, Сброс по включению питания, Сброс от BOD- схемы, Сброс от WDT. |
| 2 | 0x0001 | INT0 | Запрос от Внешнего Прерывания на выводе INT0 |
| 3 | 0x0002 | INT1 | Запрос от Внешнего Прерывания на выводе INT1 |
| 4 | 0x0003 | TIMER1 CAPT | Таймер/Счетчик1 в режиме Захвата |
| 5 | 0x0004 | TIMER1 COMPA | Таймер/Счетчик1 в Режиме Сравнения на равенство TCNT1 и OCR1A |
| 6 | 0x0005 | TIMER1 OVF | Переполнение Таймера/Счетчика 1 |
| 7 | 0x0006 | TIMER0 OVF | Переполнение Таймера/Счетчика 0 |
| 8 | 0x0007 | USART0, RX | Чтение через USART0 завершено |
| 9 | 0x0008 | USART0, UDRE | Регистр Данных UART 0 пуст |
| 10 | 0x0009 | UART0, TX | Передача через USART0 завершено |
| 11 | 0x000A | ANALOG COMP | Прерывание от Аналогового Компаратора |
| 12 | 0x000B | PCINT | Прерывание на выводах PCINT7..0 |
| 13 | 0x000C | TIMER1 COMPB | Таймер/Счетчик1 в Режиме Сравнения на равенство TCNT1 и OCR1B |
| 14 | 0x000D | TIMER0 COMPA | Таймер/Счетчик 0 в Режиме Сравнения на равенство TCNT0 и OCR0A |
| 15 | 0x000E | TIMER0 COMPB | Таймер/Счетчик0 в Режиме Сравнения на равенство TCNT0 и OCR0B |
| 16 | 0x00F | USI START | Условие Старта(Начала) интерфейса USI |
| 17 | 0x0010 | USI OVERFLOW | Переполнение USI |
| 18 | 0x0011 | EE READY | Память EEPROM готова к записи(т.е. выполнены условия для начала записи) |
| 19 | 0x0012 | WDT OVERFLOW | Переполнение Таймера в WDT |

Наиболее и общие программные установки для адреса Сброса и Векторов Прерывания в ATtiny2313, пример на Ассемблере:

| Номер Строки (в тексте программы он не пишется) | Адрес в Программе | команда | Имя Метки | Описание Прерывания |
|---|---|---|---|---|
| 1 | 0x0000 | rjmp | RESET | ;Прерывание на внешнем выводе ;“RESET”, Сброс по включению ;питания, Сброс от BOD- схемы, ;Сброс от WDT. |
| 2 | 0x0001 | rjmp | INT0 | ;Запрос от Внешнего Прерывания на ;выводе INT0 |
| 3 | 0x0002 | rjmp | INT1 | ;Запрос от Внешнего Прерывания на ;выводе INT1 |
| 4 | 0x0003 | rjmp | TIMER1_CAPT | ;Таймер/Счетчик1 в режиме Захвата |
| 5 | 0x0004 | rjmp | TIMER1_COMPA | ;Таймер/Счетчик1 в Режиме Сравнения на равенство TCNT1 и OCR1A |
| 6 | 0x0005 | rjmp | TIMER1_OVF | ;Переполнение Таймера/Счетчика1 |
| 7 | 0x0006 | rjmp | TIMER0_OVF | ;Переполнение Таймера/Счетчика0 |
| 8 | 0x0007 | rjmp | USART0_RX | ;Чтение через USART0 завершено |
| 9 | 0x0008 | rjmp | USART0_UDRE | ;Регистр Данных UART0 пуст |
| 10 | 0x0009 | rjmp | UART0_TX | ;Передача через USART0 завершена |
| 11 | 0x000A | rjmp | ANALOG_COMP | ;Прерывание от Аналогового ;Компаратора |
| 12 | 0x000B | rjmp | PCINT | ;Прерывание на выводах PCINT |
| 13 | 0x000C | rjmp | TIMER1_COMPB | ;Таймер/Счетчик1 в Режиме Сравнения на равенство TCNT1 и OCR1B |
| 14 | 0x000D | rjmp | TIMER0_COMPA | ;Таймер/Счетчик0 в Режиме Сравнения на равенство TCNT0 и OCR0A |
| 15 | 0x000E | rjmp | TIMER0_COMPB | ;Таймер/Счетчик0 в Режиме Сравнения на соответствие B |
| 16 | 0x00F | rjmp | USI_START | ;Условие Старта(Начала) интерфейса ;USI |
| 17 | 0x0010 | rjmp | USI_OVERFLOW | ;Переполнение USI |
| 18 | 0x0011 | rjmp | EE_READY | ;Память EEPROM готова к записи(т.е. ;выполнены условия для начала ;записи) |
| 19 | 0x0012 | rjmp | WDT_OVERFLOW | ;Переполнение Таймера в WDT |
| ; Дальнейший текст в программе | | | | |
| | 0x0013 | RESET: | ldi r16,low(RAMEND) | ;начало Главной программы |
| | 0x0014 | | out SPL,r16 | ;Установить Указатель Стека ;Программ на Верхний Адрес в SRAM |
| | 0x0015 | | sei | ;Установить Глобальный Бит ;Разрешения Прерываний |
| | 0x0016 | и так | далее… | |
| | ……. | …… | ……. | |

# I/O-Порты (Порты ввода/вывода)

## Вступление

Все AVR-порты могут работать с функциями Чтения-Модификации-Записи значений на их выводах, если они используются как Общие Цифровые I/O-порты. Это значит, что направление ввода/вывода информации может быть изменено на каждом отдельном выводе, без риска неумышленного изменения направления ввода/вывода у другого вывода порта, также, имеется возможность проверки наличия какого-либо логического уровня на каждом отдельном выводе путем применения SBI- и CBI-инструкций. Та же самая ситуация и при установке какого-либо логического уровня на отдельном выводе (если он конфигурируется как Выход) или при подключении/отключении Подтягивающих Резисторов (если вывод настраивается как Вход). Все токовые буферы выводов имеют идентичные характеристики и могут поглощать или обеспечивать в нагрузке относительно большой ток. Драйверы выводов являются достаточно сильными, чтобы обеспечить управление светодиодными индикаторами. Все выводы портов имеют индивидуально подключаемые подтягивающие резисторы, которые, в свою очередь, уже подключены к источнику Vcc. Все I/O-выводы имеют защитные диоды, подключенные к Vcc и GND, это показано на Рисунке 21. Обратитесь к разделу "Электрические Характеристики" на странице 176 для ознакомления с полным списком параметров.

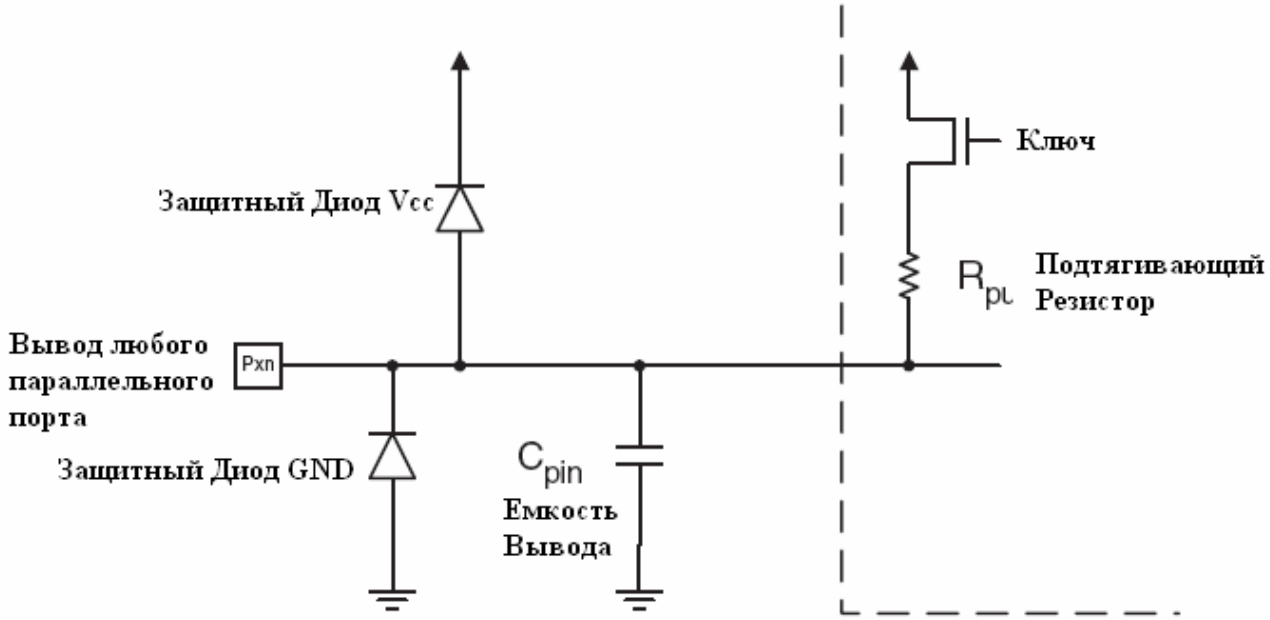



Рисунок 21 – Схема I/O-портов

Все Регистры и Биты в этом разделе имеют только общее описание. Строчная буква "x" представляет собой номер (точнее Имя порта: A,B,C и т.д.), а строчная буква "n" представляет собой номер Бита Порта. Однако, при определении Регистров и Выводов, в тексте программы должны быть использованы их настоящие имена. Например, "PORTB3" это Бит 3 в порте B (PortB); здесь используется общий вид: "PORTxn". О физическом расположении Регистров ввода/вывода и их Битов рассказано в разделе "Описание Регистров I/O-портов" на странице 57.

Для каждого I/O-порта имеется три адресуемой области памяти, которая представляет собой :1. Регистр Данных - "PORTx"; 2. Регистр для настройки Направления Данных - "DDRx"; 3. Бит-"PINx"; Бит-"PINx" разрешается только читать, а вот Регистр Данных - "PORTx" и Регистр для настройки Направления Данных - "DDRx" может быть и прочитан и записан. Однако, запись логической единицы в Бит-"PINx" дает в результате переключение соответствующего бита в Регистре Данных "PORTx". Плюс ко всему, имеется возможность отключения всех Подтягивающих Резисторов на всех выводах всех портов только лишь одной установкой бита PUD в регистре MCUCR.

Как использовать I/O-порты для Общего Цифрового Ввода/Вывода описано в разделе "I/O-порты для Общего Цифрового Ввода/Вывода" на странице 46. Большинство выводов портов совмещают альтернативные функции, обеспечивая работу периферийных устройств МК. О том, как каждая альтернативная функция взаимодействует со своим выводом, написано в разделе "Альтернативные Функции Портов" на странице 50. Обратитесь к разделам, описывающим конкретные модули МК, для получения полной информации об альтернативных функциях.

Обратите внимание, что разрешение альтернативных функций у соответствующих выводов, никак не воздействует и не мешает работать другим выводам порта и выполнять функцию Общего ввода/вывода.

# I/O-порты для Общего Цифрового Ввода/Вывода

Эти параллельные порты являются двунаправленными. На Рисунке 22 приведена функциональная схема одного I/O-вывода, который здесь назван обобщающим именем Pxn.

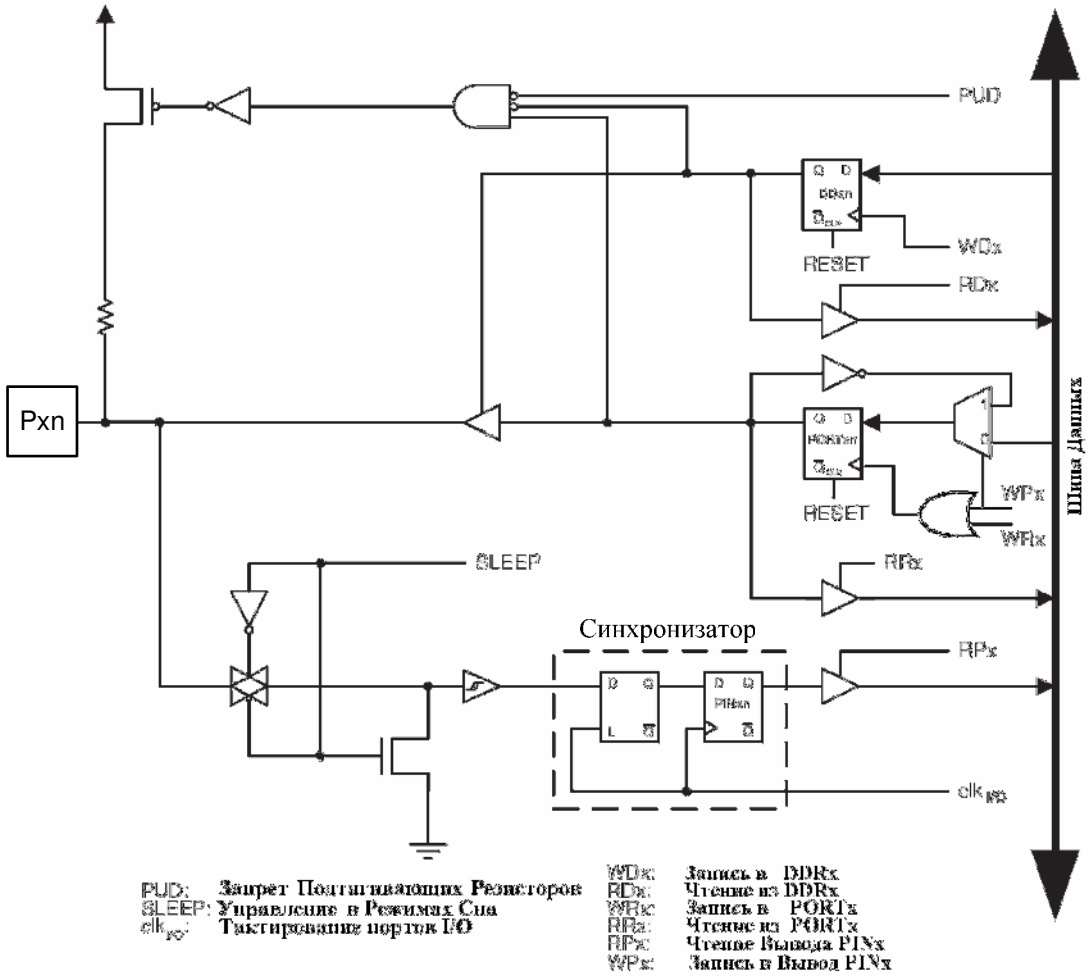



Рисунок 22 – Вывод для Общего Цифрового Ввода/Вывода

Примечание: 1. Обозначения WRx, WPx, WDx, RRx, RPx, и RDx являются общими для вех выводов в пределах одного порта, а $clk_{I/O}$, SLEEP, и PUD являются общими для всех портов.

# Конфигурирование Выводов

Каждый порт имеет три вида битов, каждый бит относится к своему регистру; виды битов: DDxn, PORTxn, и PINxn биты. Как рассказано в разделе "Описание Регистров I/O-портов" на странице 57, DDxn биты относятся к регистру DDRx, биты PORTxn относятся к регистру PORTx, а биты PINxn относятся к регистру PINx.

Биты DDxn регистра DDRx определяют направление передачи данных через выводы порта. Если DDxn бит записан единицей, тогда вывод Pxn сконфигурирован как Выход. А, если DDxn бит записан нулем, тогда вывод Pxn сконфигурирован как Вход.

Если в биты PORTxn записать логическую единицу, когда выводы сконфигурированы как Входы, то к выводам подключатся подтягивающие резисторы. Для отключения этих резисторов, в биты PORTxn надо записать логические нули или же сконфигурировать выводы как Выходы. Выводы любого порта будут находиться в третьем состоянии, если произойдет сброс МК или же, если не будет работать Тактовый Генератор .

Если же в биты PORTXn записать логическую единицу, когда выводы сконфигурированы как Выходы, то на выводах будет установлен высокий логический уровень. А, если в биты PORTxn записать логические нули, когда выводы сконфигурированы как Выходы, то на каждом выводе будет низкий логический уровень.

# Настройка Выводов

Запись логической единицы в бит PINxn меняет значение в регистре PORTxn, в зависимости от битов DDRxn. Обратите внимание, что инструкция SBI может устанавливать единицу только в один бит определенного порта.

# Переключение между состоянием Ввода и Вывода

Возможны такие состояния:

№1. "Третье" или "Z-состояние", когда ({DDxn, PORTxn} = 0b00), выводы являются Входами и имеют большое входное сопротивление;

№2. Выводы являются Выходами. На Выходах установлены высокие логические уровни, когда ({DDxn, PORTxn} = 0b11);

№3. Состояние, когда выводы являются Входами и к ним подключаются подтягивающие резисторы, такая установка битов: {DDxn, PORTxn} = 0b01) ;

№4. Состояние, когда выводы являются Выходами и на них установлены низкие логические уровни, в этом случае биты настраиваются так: ({DDxn, PORTxn} = 0b10);

Обычно состояние, когда выводы являются Входами и к ним подключаются подтягивающие резисторы чаще всего приемлемо в большинстве приложений, т.к. к выводам часто подключают внешние сопротивления с большими номиналами и притягивают их к Vcc. Если, все-таки, этот случай не подходит, то подтягивающие резисторы можно отключить установкой PUD-бита в Регистре MCUCR.

При переходе из Третьего состояния №1 в состояние №2, надо использовать состояние №3 или №4 в качестве промежуточного.

При переключении между состояниями №3 и №4 возникают одни и те же проблемы. Тут пользователь должен использовать остальные состояния (какое-нибудь одно) №1 или №2 в качестве промежуточных.

В Таблице 22 показаны биты для настройки портов и значений на них.

**Таблица 22 – Конфигурация выводов параллельных портов**

| DDxn | PORTxn | PUD-бит (в регистре MCUCR) | Порт-I/O | Pull-up-резисторы (подтягивающие) | Комментарий |
|---|---|---|---|---|---|
| 0 | 0 | X | Вход | Нет | «Третье Состояние» (Z-состояние) |
| 0 | 1 | 0 | Вход | Да | Pxn-вывод будет источником тока, если его нагрузка подключена на землю |
| 0 | 1 | 1 | Вход | Нет | «Третье Состояние» (Z-состояние) |
| 1 | 0 | X | Выход | Нет | На Выходе низкий логический уровень (вывод является нагрузкой) |
| 1 | 1 | X | Выход | Нет | На Выходе высокий логический уровень (вывод является источником тока) |

# Чтение Значений на Выводах

Независимо от настроек регистров DDxn, выводы портов могут быть прочитаны через биты PINx. Как показано на **Рисунке 22**, вывод PINxn вместе с Фиксатором образуют Синхронизатор. Это помогает избежать нестабильности, если уровень сигнала на выводе изменяется рядом с фронтом импульса тактовой частоты, однако, это, также, вносит дополнительную задержку. На **Рисунке 23** показаны Временные Диаграммы процесса синхронизации при чтении значения на внешнем выводе. Максимальная и Минимальная задержка обозначены как $t_{pd,max}$ и $t_{pd,min}$ соответственно.

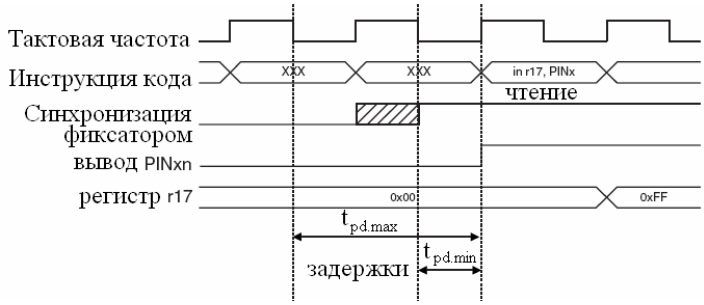



Рисунок 23 - Временные Диаграммы процесса синхронизации при чтении значения на внешнем выводе

Здесь(на Рисунке 23) рассматривается тактовый период, начинающийся после первого заднего фронта тактовой частоты (первая пунктирная вертикальная линия). Фиксатор не работает, если уровень тактового импульса является низким, когда тактовый уровень становится высоким, Фиксатор становится "прозрачным" и пропускает сигнал с вывода, этот процесс показан в заштрихованной области на Рисунке 23. Значение сигнала на выводе является недоступным вблизи заднего фронта тактового импульса. Значение на выводе читается в бит PINxn только, когда уровень тактового импульса начинает нарастать (по переднему фронту). Как показано стрелками, обозначающими временную задержку, уровень сигнала на выводе, который надо прочитать, может быть задержан от 0,5 до 1,5 периода тактовой частоты, и только потом значение сигнала на выводе будет доступно для бита PINxn, время задержки зависит от того, как скоро появится передний фронт тактового импульса.

При чтении программой пользователя значения на выводе МК необходимо использовать (вставлять) nop-инструкции в текст программы так, как это показано на Рисунке 24. Инструкция "out" устанавливает сигнал Фиксатора на передний фронт тактовой частоты. В этом случае время задержки $t_{pd}$ через Синхронизатор будет равно одному периоду тактовой частоты.

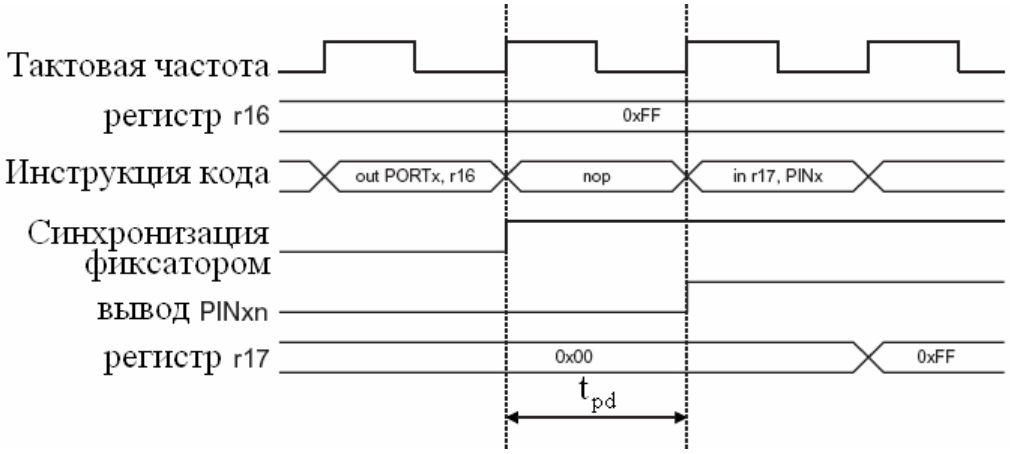



Рисунок 24 – Синхронизация при чтении программой значения на выводе МК

Следующие примеры кода показывают, как настроить порт B следующим образом:
выводы PB0 и PB1 – это Выходы с высоким уровнем сигнала на них;
выводы PB2 и PB3 – это Выходы с низким уровнем сигнала на них;
выводы PB4 и PB5 – это Входы в Третьем высокоимпедансном состоянии (Z-состоянии);
выводы PB6 и PB7 – это Входы с подключенными подтягивающими к Vcc резисторами;

Assembly Code Example(1)

```
...
; Define pull-ups and set outputs high
; Define directions for port pins
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
; Insert nop for synchronization
nop
; Read port pins
in   r16,PINB
...
```

C Code Example

```
unsigned char i;
...
/* Define pull-ups and set outputs high */
/* Define directions for port pins */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Insert nop for synchronization*/
__no_operation();
/* Read port pins */
i = PINB;
...
```

Примечание: 1.
В ассемблерном коде Временные Регистры используются для минимизации времени настройки выводов 0, 1, 6 и 7, пока биты направления ввода/вывода не установлены должным образом; выводы 2 и 3 определяются как Выходы с низким уровнем сигнала на них, а выводы 0 и 1 определяются как Выходы с высоким уровнем сигнала на них.

# Разрешение Цифрового Ввода и Режимы Сна

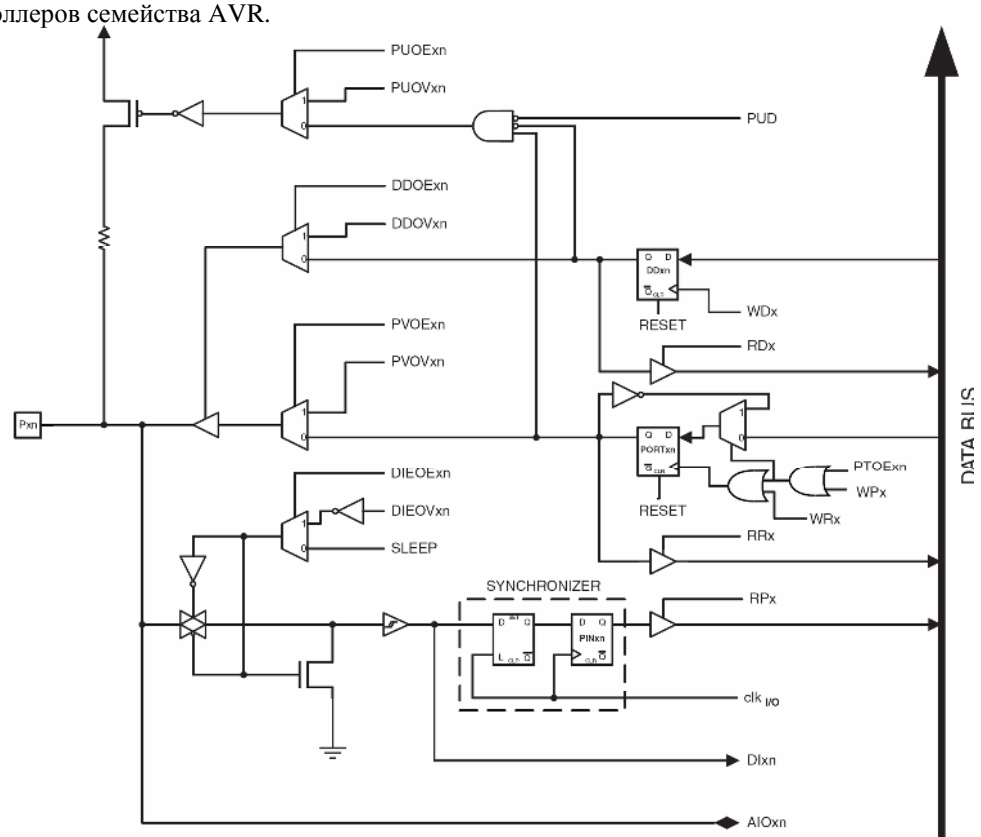
Как показано на Рисунке 22, входной цифровой сигнал может быть закорочен на GND прямо перед входом Триггера Шмитта. Сигнал, обозначенный на рисунке как SLEEP, посылается SLEEP-контроллером при входе МК в Режим Экономии Энергии или Режим Сна для того, чтобы можно было избежать большого потребления энергии на определенных выводах, если сигнал на них присутствует, но не используется или, если уровень аналогового сигнала достигает амплитуды Vcc/2.

Режим Сна не влияет и не отключает те выводы, которые используются для Внешних Прерываний. Если выводы не настроены на прием Внешних Прерываний, то для этих выводов Режим Сна (SLEEP-режим) можно применить всегда. SLEEP-режим, так же не влияет на те выводы, которые настроены на выполнение альтернативных функций, что описано в разделе "Альтернативные Функции Портов" на странице 50.

Если на выводе, настроенном на прием внешнего прерывания (настраивать можно в режим "Прерывание по нарастающему фронту", "Прерывание по спадающему фронту" или "Прерывание при смене уровня"), присутствует лог. единица, в то время, когда Внешние Прерывания запрещены, то при выходе из вышеупомянутого SLEEP-режима будет установлен Флаг (бит) соответствующего Внешнего Прерывания, так как, изменение сигнала на этом выводе фиксируется в SLEEP-режиме и при выходе из НЕГО воспроизводится заново.

# Альтернативные Функции Портов

Большинство выводов портов выполняют альтернативные функции помимо основной функции ввода/вывода цифровой информации. На Рисунке 25 показано, как Сигналы Управления, упрощенно показанные на Рисунке 22, могут быть проигнорированы Альтернативными Функциями. Игнорирование Сигналов Управления нельзя устанавливать на все выводы порта, однако Рисунок 25 является общим описанием, применимым ко всем выводам портов микроконтроллеров семейства AVR.




| | | | |
|---|---|---|---|
| Сигнал PUOExn: | Для Pxn – Разрешено Игнорирование настроек Подтягивающих Резисторов | Бит PUD: | Запрет на все Подтягивающие Резисторы |
| Сигнал PUOVxn: | Для Pxn – Сигнал, включающий Игнорирование Подтягивающих Резисторов | Сигнал WDx: | Запись в DDRx |
| Сигнал DDOExn: | Для Pxn – Разрешено Игнорирование настроек Направления в/в Данных | Сигнал RDx: | Чтение из DDRx |
| Сигнал DDOVxn: | Для Pxn – Сигнал, включающий Игнорирование настроек Направления в/в Данных | Сигнал RRx: | Чтение Регистра PORTx |
| Сигнал PVOExn: | Для Pxn – Разрешено Игнорирование выходного Значения на Выводе порта | Сигнал WRx: | Запись в Регистр PORTx |
| Сигнал PVOVxn: | Для Pxn – Сигнал, включающий Игнорирование выходного Значения на Выводе порта | Сигнал RPx: | Чтение Вывода PINx |
| Сигнал DIEOExn: | Для Pxn – Разрешено Игнорирование того, что Вывод настроен как цифровой вход | Сигнал WPx: | Запись в Вывод PINx |
| Сигнал DIEOVxn: | Для Pxn – Сигнал, включающий Игнорирование того, что Вывод настроен как цифровой вход | $clk_{I/O}$: | Такты для системы в/в |
| Сигнал SLEEP: | Для Pxn – Сигнал Управления Режимом Сна | DIxn: | Вывод Порта выполняет Альтернативную функцию Цифрового **ввода** |
| Сигнал PTOExn: | Для Pxn – Разрешение Инвертирования значения на выводе | AIOxn: | Вывод Порта выполняет Альтернативную функцию Аналогового **ввода/вывода** |

Рисунок 25 – Альтернативные Функции Портов(1)

Примечание переводчика к Рисунку 25:

1. Pxn: P – это Вывод; x – Имя Порта; n – номер Вывода;
2. DATABUS – Шина Данных;

Примечание: 1. Обозначения WRx, WPx, WDx, RRx, RPx, и RDx являются общими для вех выводов в пределах одного порта, а $clk_{I/O}$, SLEEP, и PUD являются общими для всех портов. Все остальные сигналы являются уникальными для каждого вывода.

В Таблицу 23 сведены и описаны все Сигналы Игнорирования. Индексация Портов и Выводов, показанная на Рисунке 25, не используется в Таблице 23. Все Сигналы Игнорирования вырабатываются внутри каждого модуля, имеющего Альтернативную Функцию.

**Таблица 23 – Общее описание Игнорирующих Сигналов для Альтернативных Функций**

| Название Сигнала | Полное Имя Сигнала | Описание Сигнала |
|---|---|---|
| PUOE | Pull-up Override Enable - Разрешено Игнорирование настроек Подтягивающих Резисторов | Если этот Сигнал установлен, тогда Подтягивающие Резисторы под контролем сигнала PUOV. Если PUOE отсутствует, то подтягивающие резисторы могут быть включены, естественно, если загружены соответствующие биты в соответствующие регистры: DDxn, PORTxn и PUD) |
| PUOV | Pull-up Override Value - Сигнал, включающий Игнорирование Подтягивающих Резисторов | Если PUOE сигнал установлен, тогда Подтягивающие Резисторы разрешены/запрещены, если PUOV присутствует/отсутствует, независимо от настроек DDxn, PORTxn и PUD бита. |
| DDOE | Data Direction Override Enable - Разрешено Игнорирование настроек Направления в/в Данных | Если этот Сигнал присутствует, то функционирование Драйвера Выхода (вывода) находится под контролем сигнала DDOV. Если DDOE отсутствует, то работа Драйвера Выхода может быть разрешена установкой соответствующего бита в Регистре DDxn. |
| **DDOV** | Data Direction Override Value - Сигнал, включающий Игнорирование настроек Направления в/в Данных | Если сигнал **DDOE** присутствует, то Драйвер Выхода включен/выключен, если DDOV присутствует/отсутствует независимо от настроек DDxn Регистра. |
| PVOE | Port Value Override Enable - Разрешено Игнорирование выходного Значения на Выводе порта | Если этот сигнал присутствует, и Драйвер Выхода включен, то значение в порте контролируется сигналом PVOV. Если PVOE очищен, а Драйвер Выхода по-прежнему включен, то значение на выводе порта будет зависеть от битов PORTxn. |
| PVOV | Port Value Override Value – Сигнал, включающий Игнорирование выходного Значения на Выводе порта | Если присутствует сигнал PVOE, то значение на выводе порта равно значению PVOV независимо от настроек битов PORTxn. |
| PTOE | Port Toggle Override Enable – Разрешение Инвертирования значения на выводе | Если присутствует сигнал PTOE, то бит в PORTxn инвертируется. |
| DIEOE | Digital Input Enable Override Enable – Разрешено Игнорирование того, что Вывод настроен как цифровой вход | Если этот сигнал установлен, то Разрешение Цифрового Ввода контролируется сигналом DIEOV. Если DIEOE отсутствует, то Разрешение Цифрового Ввода информации определяется тем, в каком режиме находится МК – в Нормальном или каком-нибудь режиме сна. |
| DIEOV | Digital Input Enable Override Value – Сигнал, включающий Игнорирование того, что Вывод настроен как цифровой вход | Если сигнал DIEOE присутствует, то Цифровой Ввод разрешен/запрещен, если сигнал DIEOV присутствует/отсутствует независимо от того, в каком режиме работает МК – в режиме сна или Нормальном. |
| DI | Digital Input – Цифровой Ввод | Эта функция Цифрового Ввода является альтернативной функцией. Как показано на Рисунке 25, этот сигнал подается на выход Триггера Шмитта перед Синхронизатором. Если Цифровой Ввод используется в качестве источника тактирования, то модуль, имеющий Альтернативную Функцию, будет использовать этот источник для собственной синхронизации. |
| AIO | Аналоговый ввод/вывод | Функция AIO так же является Альтернативной. Сигнал подводится/снимается непосредственно с вывода, этот вывод может использоваться как двунаправленный вывод. |

Следующие подразделы кратко описывают Альтернативные Функции для каждого порта и описывают соответствие между Игнорирующими Сигналами и Альтернативными Функциями. Обратитесь к разделам, описывающим Альтернативные Функции, за более детальной информацией.

# MCUCR – Регистр Управления МК

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | PUD | SM1 | SE | SM0 | ISC11 | ISC10 | ISC01 | ISC00 | MCUCR |
| Чтение/ Запись | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | |
| Начальные значения | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | |

**• Bit 7 – PUD: Подтягивающие Резисторы Запрещены**

Если этот бит записан лог. единицей, то Подтягивающие Резисторы I/O портов запрещены, даже, если Регистры DDxn и PORTxn настроены и разрешают ИХ включить ({DDxn, PORTxn} = 0b01). Обратитесь к разделу «Конфигурирование Выводов» на странице 46 за более подробной информацией об этом.

# Альтернативные Функции выводов Порта А «PORTA»

**Альтернативные Функции выводов ПортаА «PORTA»показаны в Таблице 24**

**Таблица 24 – Альтернативные Функции выводов ПортаА «PORTA»**

| Выводы ПортаА | Альтернативные Функции |
|---|---|
| PA2 | RESET, dW |
| PA1 | XTAL2 |
| PA0 | XTAL1 |

# Альтернативные Функции выводов ПортаВ «PORTB»

**Альтернативные Функции выводов Порта В «PORTA»показаны в Таблице 25**

**Таблица 25 – Альтернативные Функции выводов Порта В «PORTB»**

| Выводы ПортаВ | Альтернативные Функции |
|---|---|
| PB7 | USCK/SCL/PCINT7 |
| PB6 | DO/PCINT6 |
| PB5 | DI/SDA/PCINT5 |
| PB4 | OC1B/PCINT4 |
| PB3 | OC1A/PCINT3 |
| PB2 | OC0A/PCINT2 |
| PB1 | AIN1/PCINT1 |
| PB0 | AIN0/PCINT0 |

Альтернативные Выводы сконфигурированы так, как показано далее:

**• USCK/SCL/PCINT7 – Порт В, Бит 7**

**Альтернативная Функция USCK**: В трехпроводном режиме вывод является синхронизирующим для Универсального Последовательного Интерфейса (USI).

**Альтернативная Функция SCL**: В двухпроводном режиме: вывод является синхронизирующим для двухпроводного режима интерфейса USI.

**Альтернативная Функция PCINT7**: Вывод, устанавливается в качестве источника прерывания. Вывод PB7 может служить источником Внешнего Прерывания.

**• DO/PCINT6 – Порт В, Бит 6**

**Альтернативная Функция DO**: В трехпроводном режиме: это вывод Выходных Данных из интерфейса USI. В трехпроводном режиме Выходные Данные игнорируют значение на выводе PORTB6 и передаются в порт, если бит направления данных DDB6 установлен в 1. Однако, бит PORTB6 все еще контролирует настройку Подтягивающих Резисторов, если вывод является Входом, а в бит PORTB6 загружена 1.

**Альтернативная Функция PCINT6**: Вывод, устанавливается в качестве источника прерывания. Вывод PB6 может служить источником Внешнего Прерывания.

**• DI/SDA/PCINT5 – Порт B, Бит 5**
**Альтернативная Функция DI**: В трехпроводном режиме вывод является Входом для Данных USI-интерфейса. В этом режиме нормальные функции порта не игнорируются, поэтому вывод должен быть сконфигурирован как Вход.
**Альтернативная Функция SDA**: В двухпроводном режиме вывод используется для работы с данными USI-интерфейса.
**Альтернативная Функция PCINT5**: Вывод, устанавливается в качестве источника прерывания. Вывод PB5 может служить источником Внешнего Прерывания.

**• OC1B/PCINT4 – Порт B, Бит 4**
**Альтернативная Функция OC1B**: Этот ножка служит для вывода данных при работе в Режиме Совпадения B: Ножка PB4 может служить внешним выходом для Таймера/Счетчика 1, работающего в Режиме Совпадения B. Для того, чтобы воспользоваться этой функцией, этот вывод МК нужно настроить как Выход (загрузив в DDB6 число 1.
**Альтернативная Функция PCINT4:** Вывод, устанавливается в качестве источника прерывания. Вывод PB4 может служить источником Внешнего Прерывания.

**• OC1A/PCINT3 – Порт B, Бит 3**
**Альтернативная Функция OC1A**: Этот ножка служит для вывода данных при работе в Режиме Совпадения B: Ножка PB3 может служить внешним выходом для Таймера/Счетчика 1, работающего в Режиме Совпадения A. Для того, чтобы воспользоваться этой функцией, этот вывод МК нужно настроить как Выход (загрузив в DDB3 число 1).
**Эта ножка, также, используется как выход, если Таймер/Счетчик 1 работает в режиме ШИМ.**
**Альтернативная Функция PCINT3:** Вывод, устанавливается в качестве источника прерывания. Вывод PB3 может служить источником Внешнего Прерывания.

**• OC0A/PCINT2 – Порт B, Бит 2**
**Альтернативная Функция OC0A**: Этот ножка служит для вывода данных при работе в Режиме Совпадения B: Ножка PB2 может служить внешним выходом для Таймера/Счетчика 0, работающего в Режиме Совпадения A. Для того, чтобы воспользоваться этой функцией, этот вывод МК нужно настроить как Выход (загрузив в DDB2 число 1).
**Эта ножка, также, используется как выход, если Таймер/Счетчик 0 работает в режиме ШИМ.**
**Альтернативная Функция PCINT2:** Вывод, устанавливается в качестве источника прерывания. Вывод PB2 может служить источником Внешнего Прерывания.

**• AIN1/PCINT1 – Порт B, Бит 1**
**Альтернативная Функция AIN1:** Выполняя эту функцию, эта ножка является Инвертирующим Входом Аналогового Компаратора. Чтобы избежать вмешательства в работу Аналогового Компаратора Функцией Цифрового ввода, необходимо настроить этот вывод как вход с **отключенным** подтягивающим резистором.
**Альтернативная Функция PCINT1:** Вывод, устанавливается в качестве источника прерывания. Вывод PB1 может служить источником Внешнего Прерывания.

**• AIN0/PCINT0 – Порт B, Бит 0**
**Альтернативная Функция AIN0:** Выполняя эту функцию, эта ножка является НеИнвертирующим Входом Аналогового Компаратора. Чтобы избежать вмешательства в работу Аналогового Компаратора Функцией Цифрового ввода, необходимо настроить этот вывод как вход с **отключенным** подтягивающим резистором.
**Альтернативная Функция PCINT0:** Вывод, устанавливается в качестве источника прерывания. Вывод PB0 может служить источником Внешнего Прерывания.

В Таблицах 26 и 27 показано, какая существует связь между Альтернативными Функциями Порта B и Игнорирующими Сигналами, которые показаны на Рисунке 25 (страница 50). Сигналы SPI MSTR INPUT и SPI SLAVE OUTPUT составляют один MISO-сигнал в то время, как сигнал MOSI делится на сигнал SPI MSTR OUTPUT и сигнал SPI SLAVE INPUT.

**Таблица 26 – Игнорирующие Сигналы для Альтернативных Функций выводов PB7…PB4**

| Имя Сигнала | PB7/USCK/SCL/ PCINT7 | PB6/DO/ PCINT6 | PB5/SDA/DI/ PCINT5 | PB4/OC1B/ PCINT4 |
|---|---|---|---|---|
| PUOE | 0 | 0 | 0 | 0 |
| PUOV | 0 | 0 | 0 | 0 |
| DDOE | USI_TWO_WIRE | 0 | USI_TWO_WIRE | 0 |
| DDOV | (USI_SCL_HOLD+ $\overline{\text{PORTB7}}$·DDB7) | 0 | ( $\overline{\text{SDA}}$ + $\overline{\text{PORTB5}}$ )· ·DDRB5) | 0 |
| PVOE | (USI_TWO_WIRE·DDRB7) | USI_TREE_WIRE | (USI_TWO_WIRE·DDRB5) | OC1B_PVOE |
| PVOV | 0 | DO | 0 | OC1B_PVOV |
| PTOE | USI_PTOE | 0 | 0 | 0 |
| DIEOE | (PCINT7·PCIE+USISIE) | (PCINT6·PCIE) | (PCINT5·PCIE)+USISIE | (PCINT4·PCIE) |
| DIEOV | 1 | 1 | 1 | 1 |
| DI | PCINT7 INPUT<br>USCK INPUT<br>SCL INPUT | PCINT6 INPUT | PCINT5 INPUT<br>SDA INPUT<br>DI INPUT | PCINT4 INPUT |
| AIO | - | - | - | - |

USI_TWO_WIRE – двухпроводной режим USI;
USI_TREE_WIRE – трехпроводной режим USI;
HOLD – удержание;

**Таблица 27 – Игнорирующие Сигналы для Альтернативных Функций выводов PB3…PB0**

| Имя Сигнала | PB3/OC1A/ PCINT3 | PB2/OC0A/ PCINT2 | PB1/AIN1/ PCINT1 | PB0/AIN0/ PCINT0 |
|---|---|---|---|---|
| PUOE | 0 | 0 | 0 | 0 |
| PUOV | 0 | 0 | 0 | 0 |
| DDOE | 0 | 0 | 0 | 0 |
| DDOV | 0 | 0 | 0 | 0 |
| PVOE | OC1A_PVOE | OC0A_PVOE | 0 | 0 |
| PVOV | OC1A_PVOV | OC0A_PVOV | 0 | 0 |
| PTOE | 0 | 0 | 0 | 0 |
| DIEOE | (PCINT3 PCIE) | (PCINT2 PCIE) | (PCINT1 PCIE) | (PCINT0 PCIE) |
| DIEOV | 1 | 1 | 1 | 1 |
| DI | PCINT7 INPUT | PCINT6 INPUT | PCINT5 INPUT | PCINT4 INPUT |
| AIO | - | - | AIN1 | AIN0 |

# Альтернативные Функции Порта D

**Альтернативные Функции выводов Порта D «PORTD»показаны в Таблице 28**

**Таблица 28 – Альтернативные Функции выводов Порта D**

| Выводы Порта D | Альтернативные Функции |
|---|---|
| PD6 | ICP |
| PD5 | OC0B/T1 |
| PD4 | T0 |
| PD3 | INT1 |
| PD2 | INT0/XCK/CKOUT |
| PD1 | TXD |
| PD0 | RXD |

**• ICP – Порт D, Бит 6**
**Альтернативная Функция ICP**: Эта ножка является Входом для Захвата Таймером/Счетчиком1.
**• OC0B/T1 – Порт D, Бит 5**
**Альтернативная Функция OC0B**: Этот ножка служит для вывода данных при работе в Режиме Совпадения B: Ножка PB2 может служить внешним выходом для Таймера/Счетчика 0, работающего в Режиме Совпадения B. Для того, чтобы воспользоваться этой функцией, этот вывод МК нужно настроить как Выход (загрузив в DDB5 бит 1).
**Эта ножка, также, используется как выход, если Таймер/Счетчик 0 работает в режиме ШИМ.**
**Альтернативная Функция T1**: Внешний Тактовый Вход Таймера/Счетчика1 включается загрузкой «1» в биты CS02 и CS01в Регистр Управления Таймером/Счетчиком1 – «TCCR1»

**• T0 – Порт D, Бит 4**
**Альтернативная Функция T0**: Внешний Тактовый Вход Таймера/Счетчика0 включается загрузкой «1» в биты CS02 и CS01в Регистр Управления Таймером/Счетчиком0 – «TCCR0»

**• INT1 – Порт D, Бит 3**
**Альтернативная Функция INT1** Источник Внешнего Прерывания 1. Вывод PD3 может служить как Источник Внешнего Прерывания МК.

**• INT0/XCK/CKOUT – Порт D, Бит 2**
**Альтернативная Функция INT0**: Источник Внешнего Прерывания 0. Вывод PD2 может служить как Источник Внешнего Прерывания МК.
**Альтернативная Функция XCK**: Синхронный Вывод, используется для синхронизации интерфейса USART, используется только в Синхронном Режиме.
**Альтернативная Функция CKOUT**: Ножка используется для вывода Тактовой Частоты МК.

**• TXD – Порт D, Бит 1**
**Альтернативная Функция TXD**: Ножка используется для **передачи** данных через интерфейс UART.

**• RXD – Порт D, Бит 0**
**Альтернативная Функция RXD**: Ножка используется для **приема** данных через интерфейс UART.

В Таблицах 29 и 30 показано, какая существует связь между Альтернативными Функциями Порта D и Игнорирующими Сигналами, которые показаны на Рисунке 25 (страница 50).

**Таблица 29 - Игнорирующие Сигналы для Альтернативных Функций выводов PD7…PD4**

| Название Сигнала | PD6/ICP | PD5/OC1B/T1 | PD4/T0 |
|---|---|---|---|
| PUOE | 0 | 0 | 0 |
| PUOV | 0 | 0 | 0 |
| DDOE | 0 | 0 | 0 |
| DDOV | 0 | 0 | 0 |
| PVOE | 0 | OC1B_PVOE | 0 |
| PVOV | 0 | OC1B_PVOV | 0 |
| PTOE | 0 | 0 | 0 |
| DIEOE | ICP разрешен | T1 разрешен | T0 разрешен |
| DIEOV | 1 | 1 | 1 |
| DI | ICP – вход | T1 – вход | T0 – вход |
| AIO | - | - | AIN1 |

**Таблица 30 - Игнорирующие Сигналы для Альтернативных Функций выводов PD3…PD0**

| Название Сигнала | PD3/INT1 | PD2/INT0B/XCK/CKOUT | PD1/TXD | PD0/RXD |
|---|---|---|---|---|
| PUOE | 0 | 0 | TXD_OE | RXD_OE |
| PUOV | 0 | 0 | 0 | $PORD0 \cdot \overline{PUD}$ |
| DDOE | 0 | 0 | TXD_OE | RXD_EN |
| DDOV | 0 | 0 | 1 | 0 |
| PVOE | 0 | XCK_PVOE | TXD_OE | 0 |
| PVOV | 0 | XCKO_PVOV | TXD_PVOV | 0 |
| PTOE | 0 | 0 | 0 | 0 |
| DIEOE | INT1 разрешен | INT0 разрешен / XCK разрешен как Вход | 0 | 0 |
| DIEOV | 1 | 1 | 0 | 0 |
| DI | INT1 настроен как Вход | INT0 как Вход/ XCK как Вход | 0 | RXD как Вход |
| AIO | - | - | - | - |

# Описание Регистров для I/O-портов

## Регистр Порт А – «PORTA»

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | PORTA2 | PORTA1 | PORTA0 | PORTA |
| Чтение/ Запись | Ч | Ч | Ч | Ч | Ч | Ч/З | Ч/З | Ч/З | |
| Начальные значения | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

## Регистр устанавливающий направление в/в данных Порта А – «DDRA»

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | DDA2 | DDA1 | DDA0 | DDRA |
| Чтение/ Запись | Ч | Ч | Ч | Ч | Ч | Ч/З | Ч/З | Ч/З | |
| Начальные значения | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

## Адрес чтения данных с выводов Порта А – «PINA»

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | PINA2 | PINA1 | PINA0 | PINA |
| Чтение/ Запись | Ч | Ч | Ч | Ч | Ч | Ч/З | Ч/З | Ч/З | |
| Начальные значения | неопр. | неопр. | неопр. | неопр. | неопр. | неопр. | неопр. | неопр. | |

## Регистр Данных Порта В – «PORTB»

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | PORTB7 | PORTB6 | PORTB5 | PORTB4 | PORTB3 | PORTB2 | PORTB1 | PORTB0 | PORTB |
| Чтение/ Запись | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | |
| Начальные значения | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

## Регистр устанавливающий направление в/в данных Порта B – «DDRB»

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | DDB7 | DDB6 | DDB5 | DDB4 | DDB3 | DDB2 | DDB1 | DDB0 | DDRB |
| Чтение/ Запись | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | |
| Начальные значения | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

## Адрес чтения данных с выводов Порта B – «PINB»

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | PINB7 | PINB6 | PINB5 | PINB4 | PINB3 | PINB2 | PINB1 | PINB0 | PINB |
| Чтение/ Запись | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | |
| Начальные значения | неопр. | неопр. | неопр. | неопр. | неопр. | неопр. | неопр. | неопр. | |

## Регистр Данных Порта D – «PORTD»

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | --- | PORTD6 | PORTD5 | PORTD4 | PORTD3 | PORTD2 | PORTD1 | PORTD0 | PORTD |
| Чтение/ Запись | Ч | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | |
| Начальные значения | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

## Регистр устанавливающий направление в/в данных Порта D – «DDRD»

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | --- | DDD6 | DDD5 | DDD4 | DDD3 | DDD2 | DDD1 | DDD0 | DDRD |
| Чтение/ Запись | Ч | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | |
| Начальные значения | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

## Адрес чтения данных с выводов Порта D – «PIND»

| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | --- | PIND6 | PIND5 | PIND4 | PIND3 | PIND2 | PIND1 | PIND0 | PIND |
| Чтение/ Запись | Ч | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | |
| Начальные значения | неопр. | неопр. | неопр. | неопр. | неопр. | неопр. | неопр. | неопр. | |

# Внешние Прерывания

Внешние прерывания вызываются выводами INT0 и INT1 или любым из выводов PCINT7..0. Заметьте, что, если прерывание РАЗРЕШЕНО, то ОНО будет вызвано даже, если соответствующий вывод настроен как Выход. Это обеспечивает возможность вызывать прерывание программным способом. Регистр Управления "PCMSK" определяет, какой из выводов PCINT7..0 может вызывать прерывание. Выводы PCINT7..0 работают независимо от Тактового Генератора МК, т.е. их работа с ним не синхронизируется, а это значит прерывание на этих выводах может быть использовано для пробуждения из разных режимов сна, кроме Холостого-«Idle».

Прерывания на выводах INT0 и INT1 могут быть вызваны Спадающим фронтом сигнала, Нарастающим фронтом или Сменой Уровня Сигнала на ножке. Эти настройки отражены в спецификации на "Регистр Управления Внешними Прерываниями А" - "EICRA". Если на выводах INT0 и INT1 разрешены прерывания и, если прерывания должны срабатывать по Уровню Сигнала, то Прерывания будут вызываться так долго, пока на выводах будет удерживаться Низкий Уровень Сигнала (лог. 0). Обратите внимание, что распознавание Спадающего Фронта и Нарастающего фронта сигнала на выводах прерывания INT0 и INT1 требует наличия Синхронизации "Модуля в/в" с Тактовым (Системным) Генератором МК, что показано на странице 21 в разделе "Описание Системного Генератора". Прерывания по Низкому Уровню на выводах INT0 и INT1 вызываются Асинхронно, а это значит прерывание на этих выводах может быть использовано для пробуждения из разных режимов сна, кроме Холостого-«Idle». Тактирование I/O-модуля приостанавливается во всех режимах сна, кроме Холостого-Idle.

Обратите внимание, что, если Прерывание по Уровню используется для пробуждения МК из Режима Экономии Энергии - "Power Down" - то этот Уровень должен удерживаться на выводе, пока МК не закончит пробуждение, только после этого будет произведен переход на Вектор Прерывания. Если Уровень Сигнала Прерывания исчезнет до того, как фиксированное Время Запуска МК истечет, то МК все равно выйдет из режима сна, но Прерывание по соответствующему Вектору вызвано не будет. Время Запуска МК определяется Битами Перемычками SUT- и CKSEL-битами, как описано в разделе "Системный Тактовый Генератор и его Настройка" на странице 21.

# MCUCR – Регистр управления МК (микроконтроллером)

Биты Управления Считыванием Прерываний находятся в «Регистре Управления Внешними Прерываниями А» - "EICRA".

| Бит № | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | PUD | SM1 | SE | SM0 | ISC11 | ISC10 | ISC01 | ISC00 | MCUCR |
| Чтение/ Запись | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | |
| Начальные Значения | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**• Бит №3, №2 - ISC11, ISC10: для настройки прерывания INT1**

Прерывание INT1 будет вызвано на внешнем выводе INT1, если Флаг I в Регистре SREG и соответствующая Маска Прерывания будут установлены. Выбор Прерывания по Уровню или по Фронту на внешнем выводе выбирается в соответствии с Таблицей 31. Выборка значения на выводе INT1 происходит до определения фронта. Если время нарастания/спадания фронта длится больше, чем Тактовый Период МК, то Прерывание будет вызвано. Более короткие импульсы вызов Прерывания не гарантируют. Если выбрано Прерывание по Низкому Уровню на выводе, то этот Уровень должен удерживаться на выводе, пока не завершится выполнение текущей инструкции МК для того, чтобы вызов прерывания произошел.

**Таблица 31 – Режимы считывания сигнала Прерывания INT1**

| ISC11 | ISC10 | Описание |
|---|---|---|
| 0 | 0 | Прерывание на INT1 вызывается по: Низкому Уровню |
| 0 | 1 | Прерывание на INT1 вызывается по: Изменению Логического Уровня на выводе |
| 1 | 0 | Прерывание на INT1 вызывается по: Спадающему Фронту |
| 1 | 1 | Прерывание на INT1 вызывается по: Нарастающему Фронту |

**• Бит №1, №0 - ISC01, ISC00: для настройки прерывания INT0**

Прерывание INT1 будет вызвано на внешнем выводе INT0, если Флаг I в Регистре SREG и соответствующая Маска Прерывания будут установлены. Выбор Прерывания по Уровню или по Фронту на внешнем выводе выбирается в соответствии с Таблицей 32.

Выборка значения на выводе INT0 происходит до определения фронта. Если время нарастания/спадания фронта длится больше, чем Тактовый Период МК, то Прерывание будет вызвано. Более короткие импульсы вызов Прерывания не гарантируют. Если выбрано Прерывание по Низкому Уровню на выводе, то этот Уровень должен удерживаться на выводе, пока не завершится выполнение текущей инструкции МК для того, чтобы вызов прерывания произошел.

**Таблица 32 – Режимы считывания сигнала Прерывания INT0**

| ISC11 | ISC10 | Описание |
|---|---|---|
| 0 | 0 | Прерывание на **INT0** вызывается по: Низкому Уровню |
| 0 | 1 | Прерывание на **INT0** вызывается по: Изменению Логического Уровня на выводе |
| 1 | 0 | Прерывание на **INT0** вызывается по: Спадающему Фронту |
| 1 | 1 | Прерывание на **INT0** вызывается по: Нарастающему Фронту |

# Регистр Макси Прерываний – «GIMSK»

| Бит № | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | INT1 | INT0 | PCIE | --- | --- | --- | --- | --- | GIMSK |
| Чтение/ Запись | Ч/З | Ч/З | Ч/З | Ч | Ч | Ч | Ч | Ч | |
| Начальные Значения | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**• Бит №7 - INT1: Внешний Запрос на прерывание INT1 Разрешен**
Вызов Прерывания на выводе INT1 разрешен, если бит INT1 и бит I в Регистре SREG установлены в лог.1. Биты ISC11 и ISC10 в Регистре MCUCR определяют, в каком режиме будет считываться Сигнал Прерывания на выводе INT1 (Таблица 31). Сигналы на выводе INT1 будут вызывать Прерывания, даже, если вывод INT1 настроен как Выход. После вызова Прерывания, будет совершен переход на соответствующий Вектор Прерывания.

**• Бит №6 – INT0: Внешний Запрос на прерывание INT0 Разрешен**
Вызов Прерывания на выводе INT0 разрешен, если бит INT0 и бит I в Регистре SREG установлены в лог.1. Биты ISC01 и ISC00 в Регистре MCUCR определяют, в каком режиме будет считываться Сигнал Прерывания на выводе INT0 (Таблица 32). Сигналы на выводе INT0 будут вызывать Прерывания, даже, если вывод INT0 настроен как Выход. После вызова Прерывания, будет совершен переход на соответствующий Вектор Прерывания.

**• Бит №5 - PCIE: Настройка Вывода в качестве источника Прерывания Разрешена**
Прерывания на выводах PCINT7..0 разрешено, если PCIE бит установлен (в 1), и I-бит в Регистре Состояния (SREG) установлен (в 1). Любое изменение на любом из выводов PCINT7..0 сгенерирует прерывание, после чего произойдет переход на соответствующий для этих выводов Вектор Прерывания. Выводы PCINT7..0 настраиваются на прерывание индивидуально битами в Регистре PCMSK.

# Регистр Флагов Внешних Прерываний – «EIFR»

| Бит № | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | INTF1 | INTF0 | PCIF | --- | --- | --- | --- | --- | EIFR |
| Чтение/ Запись | Ч/З | Ч/З | Ч/З | Ч | Ч | Ч | Ч | Ч | |
| Начальные Значения | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**• Бит №7 - INTF1: Внешний Флаг Прерывания INT1**
Когда происходит вызов прерывания на выводе INT1, то соответствующий флаг INTF1 устанавливается в 1. Если в бит I Регистра SREG и в бит INT1 Регистра GIMSK загружена 1, то при установке этого флага МК перейдет к Соответствующему Вектору Прерывания. Этот флаг очищается аппаратно после выхода из подпрограммы обработки прерывания. Этот Флаг, также, может быть очищен записью в него лог.1 . Этот Флаг постоянно очищен, если вывод INT1 работает в Режиме Прерывания по Уровню.

**• Бит №6 – INTF0: Внешний Флаг Прерывания INT0**
Когда происходит вызов прерывания на выводе INT0, то соответствующий флаг INTF0 устанавливается в 1. Если в бит I Регистра SREG и в бит INT0 Регистра GIMSK загружена 1, то при установке этого флага МК перейдет к Соответствующему Вектору Прерывания. Этот флаг очищается аппаратно после выхода из подпрограммы обработки прерывания. Этот Флаг, также, может быть очищен записью в него лог.1 . Этот Флаг постоянно очищен, если вывод INT0 работает в Режиме Прерывания по Уровню.

**• Бит 5 - PCIF: Флаг Прерывания на выводах PCINT7..0**
При смене Логического Уровня на любом из выводов PCINT7..0 происходит вызов Прерывания, при этом устанавливается соответствующий Флаг Прерывания PCIF. Если в бит I Регистра SREG и в бит PCIE Регистра GIMSK загружена 1, то при установке этого флага МК перейдет к Соответствующему Вектору Прерывания. Этот флаг очищается аппаратно после выхода из подпрограммы обработки прерывания. Этот Флаг, также, может быть очищен записью в него лог.1.

# Регистр Маски Прерываний выводов PCINT7..0 «PCMSK»

| Бит № | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | PCINT7 | PCINT6 | PCINT5 | PCINT4 | PCINT3 | PCINT2 | PCINT1 | PCINT0 | PCMSK |
| Чтение/ Запись | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | Ч/З | |
| Начальные Значения | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**Биты №7.. №0 - PCINT7..0: Маскирующие биты выводов PCINT7..0**
Каждый бит из PCINT7..0 отвечает за свой вывод. Установка какого-нибудь бита из PCINT7..0 разрешает соответствующему I/O-выводу работать в качестве источника Прерывания, при условии, что установлен бит PCIE в GIMSK. Если какого-нибудь бит из PCINT7..0 очищен (ноль), то соответствующий I/O-вывод не будет работать в качестве источника Прерывания.

# 8-битный Таймер/Счетчик 0 с ШИМ

Таймер/Счетчик 0 - это 8-битный T/C0 (Таймер/Счетчик 0) общего назначения, с двумя "Независимыми Модулями Сравнения" и "ШИМ". Все это позволяет достичь точного выполнения программ (Режим Управления) и осуществлять Тоновую Генерацию.

Главными особенностями этого Таймера являются:

- Два Независимых Модуля Сравнения
- Дважды Буферизированные Регистры Сравнения
- Очищающий Таймер по Совпадению (Авто Перезагрузка)
- Свободный от Дрожания ШИМ с Фазовой Коррекцией
- Настраиваемый Период ШИМ
- ЧастотныйГенератор
- Три независимых источника Прерывания (TOV0, OCF0A, и OCF0B)

# Обзор

Упрощенная блок-диаграмма 8-битного Таймера/Счетчика показана на рисунке 26. За информацией о действительном размещении выводов обратитесь к разделу "Конфигурация Выводов" на странице 2. Доступные ЦПУ Регистры в/в, выделены на Рисунке 26 полужирным шрифтом/линиями. Специфические Регистры Модуля T/C0 перечислены в разделе "Описание Регистров 8-битного Таймера/Счетчика 0" на странице 72.

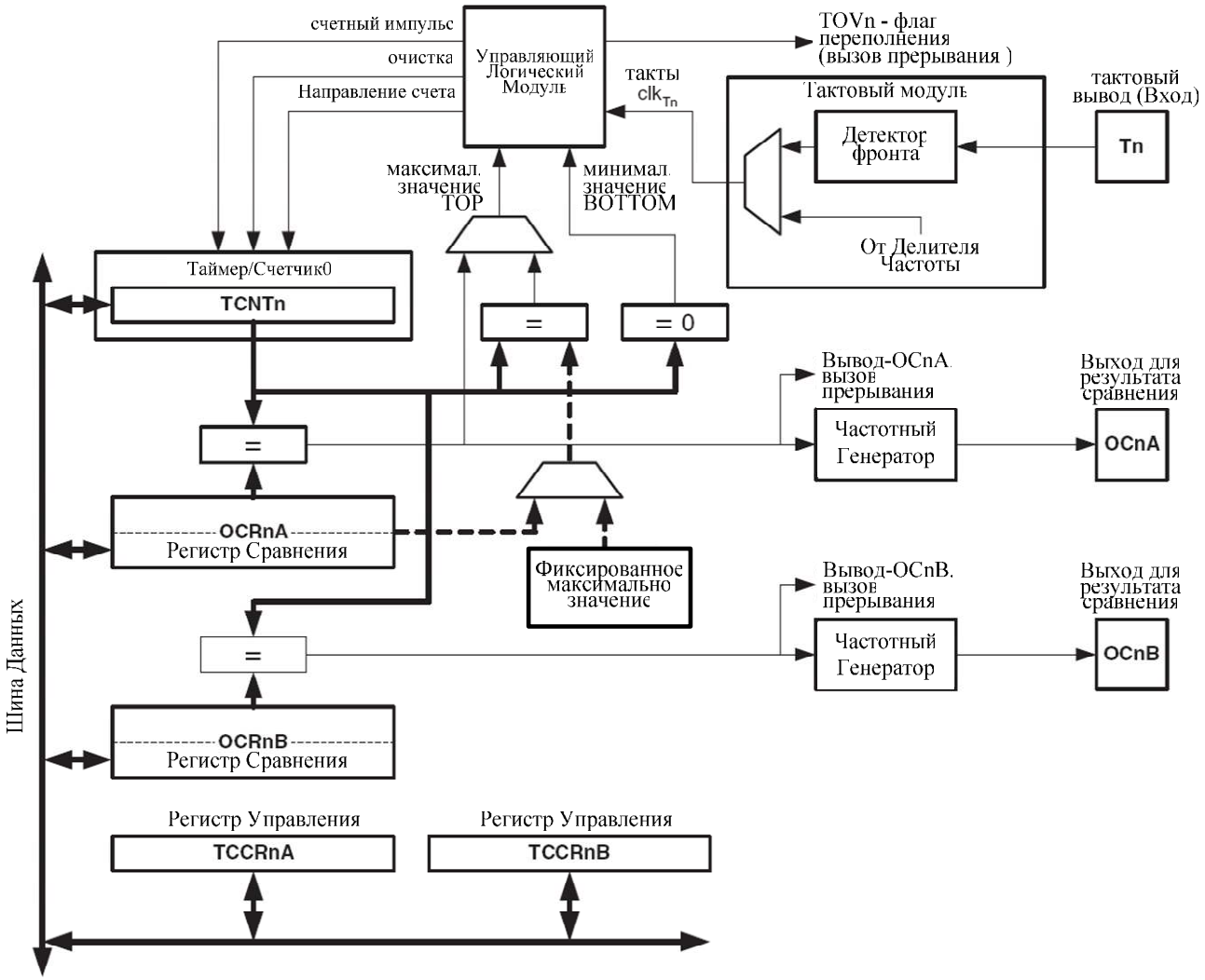



Рисунок 26 - Упрощенная блок-диаграмма 8-битного Таймера/Счетчика 0

# Регистры T/C0

Регистр TCNT0 Таймера/Счетчика0 (T/C0) и Регистры Сравнения OCR0A и OCR0B являются 8-битными. Флаги Прерываний Таймеров/Счетчиков находятся в Регистре TIFR. Также, все сигналы Прерывания Таймеров/Счетчиков имеют индивидуальные маскирующие биты в Регистре TIMSK. TIFR и TIMSK на рисунке 26 не показаны.

Таймер/Счетчик может быть тактирован от внутреннего источника через Делитель Частоты (рисунок 26) или через внешний Вывод T0. Тактовый Модуль (рисунок 26) определяет каким источником будет пользоваться T/C0 для инкрементирования на единицу (или декрементирования на единицу) своего значения (числа). T/C0 находится в неактивном состоянии, если не выбран ни один источник тактов. Выход из Тактового Модуля обозначен как $clk_{Tn}$ .

Дважды буферизированные Регистры Сравнения OCR0A и OCR0B постоянно сравниваются со значением Таймера/Счетчика0. Результат этого сравнения может быть использован Частотным Генератором для генерирования ШИМ или для изменения частоты на ножках-Выходах OC0A и OC0B.

За деталями обращайтесь к разделу "Модуль Сравнения" на странице 63. Режим Сравнения так же устанавливает Флаг Сравнения OCF0A или OCF0B, который может быть использован для вызова прерывания при совпадении сравниваемых значений.

# Определения

Многие Регистры и Биты в этом разделе описаны в общей форме. Маленькая буква "n" в названии "Timer/Counter n" обозначает номер Таймера/Счетчика, в данном случае используется Таймер/Счетчик 0. Маленькая буква "x" соответствует букве Регистра Сравнения, например, "OCR0x" означает Регистр Сравнения OCR0B или OCR0A. Однако, при использовании конкретных регистров и битов должно использоваться их настоящее имя, например, Регистр "TCNT0", который используется Таймером/Счетчиком 0 для, собственно, подсчета тактовых импульсов.

Определения, данные в Таблице 33, используются во всем документе.

Таблица 33 - Определения

| | |
|---|---|
| BOTTOM | Счетчик достигает значения BOTTOM, когда его регистр TCNT0 принимает значение 0x00. |
| MAX | Счетчик достигает значения MAX, когда его регистр TCNT0 принимает значение 0xFF (255 в десятичной с.сч.) |
| TOP | Счетчик достигает значения TOP, когда его регистр TCNT0 принимает значение, эквивалентное наивысшему допустимому значению. Значение TOP может быть установлено как 0xFF (MAX) или быть записано в Регистр Сравнения OCR0A. |

# Источники Тактирования Таймера/Счетчика 0

T/C0 может быть тактирован как внутренним источником, так и внешним. Источник тактирования выбирается Тактовым Модулем (рисунок 26-27), который, в свою очередь, настраивается битами CS02:0, которые находятся в Регистре Управления TCCR0B. За детальной информацией об Источниках Тактирования и о Делителе Частоты (Предделителе) обращайтесь к разделу "Делитель Частоты Таймера/Счетчика 0 и Таймера/Счетчика 1" на странице 79.

## Счетный Модуль

Главной частью 8-битного Таймера/Счетчика 0 является программируемый двунаправленный Счетный Модуль.

На рисунке 27 показана блок-диаграмма Счетного Модуля.

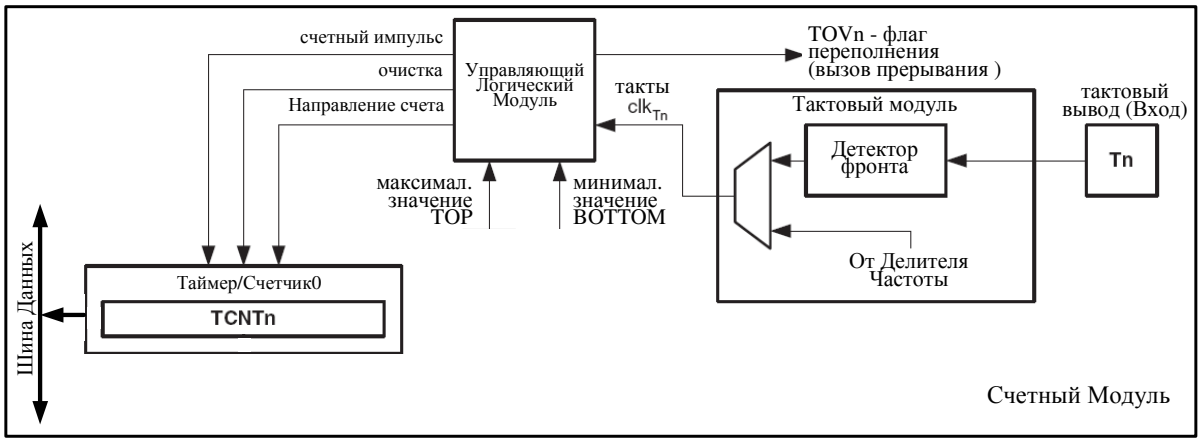



Рисунок 27 - Блок-диаграмма Счетного Модуля

Описание Сигналов на Рисунках 26-27:

- «Счетный Импульс» – выполняет Инкрементирование (+1) или Декрементирование (-1) числа в TCNT0.
- «Направления Счета» - сигнал для выбора Инкрементирования или Декрементирования.
- «Очистка» - сигнал, сбрасывающий все биты TCNT0 в нули (очистка TCNT0).
- «$clk_{Tn}$» - сигнал, тактирующий T/Cn, далее будет использоваться обозначение $clk_{T0}$.
- «TOP» - на рисунке это сигнал, сообщающий о том, что TCNT0 достиг максимального значения.
- «BOTTOM» - на рисунке это сигнал, сообщающий о том, что TCNT0 достиг минимального значения (нуля).

В зависимости от режима работы, каждый тактовый импульс "Очищает" счетчик, "Инкрементирует" или "Декрементирует" его значение. Тактовые импульсы $clk_{T0}$ могут быть получены как от внешнего, так и от внутреннего источника, это определяется битами CS02:0. Если Источник Тактов не выбран (биты установлены как CS02:0 = 0), то T/C остановлен. Однако Регистр TCNT0 может быть прочитан Процессором независимо от того, запущен счетчик или нет. Операция записи Процессором значений в счетчик имеет приоритет над операциями "Очистки" счетчика, "Инкрементирования" или "Декрементирования".

Последовательность Счета T/C определяется установкой WGM01- и WGM00-битов расположенный в Регистре Управления Таймера/Счетчика (TCCR0A) и WGM02-битом, расположенном в Регистре Управления Таймера/Счетчика B (TCCR0B). От этих битов, также, зависит, будет ли T/C0 вести себя как Счетчик или же Он будет работать в режиме Генератора Частоты. За подробной информацией о Расширенном режиме "Счетчика" и режиме "Генератора Частоты" обращайтесь к разделу "Режимы Работы" на странице 93.

Флаг Переполнения Таймера/Счетчика (TOV0) устанавливается согласно режиму работы, выбранному битами WGM01:0. Флаг TOV0 может быть использован для того, чтобы вызвать прерывание работы ЦПУ.

# Модуль Сравнения

8-битный цифровой компаратор постоянно сравнивает Регистр TCNT0 с Регистрами Сравнения OCR0A и OCR0B. В случае равенства значения, находящегося в Регистре TCNT0, с каким-нибудь Регистром - OCR0A или OCR0B, компаратор выдает Сигнал о Соответствии (Равенстве) этих регистров. Сигнал о Равенстве Регистров следующим тактом таймера устанавливает соответствующий ему Флаг Равенства (сигнализирующий бит) OCF0A или OCF0B. Если установлен Флаг Равенства и Прерывание по Совпадению значений Регистров разрешено, то будет вызвано Прерывание. Флаг Равенства очищается автоматически после выхода из Подпрограммы Обработки Прерывания. Флаг Равенства может, также, быть очищен, если выполнить команду записи в него логической единицы. Генератор Частоты использует Сигнал Равенства в соответствии со своим Режимом Работы и установками битов WGM02:0 и COM0x1:0. TOP или BOTTOM значения используются Генератором Частоты в особых случаях для обработки Критических Значений в некоторых Режимах Работы (обратитесь к разделу "Режимы Работы" на странице 93).

На рисунке 28 показана блок-диаграмма Модуля Сравнения.

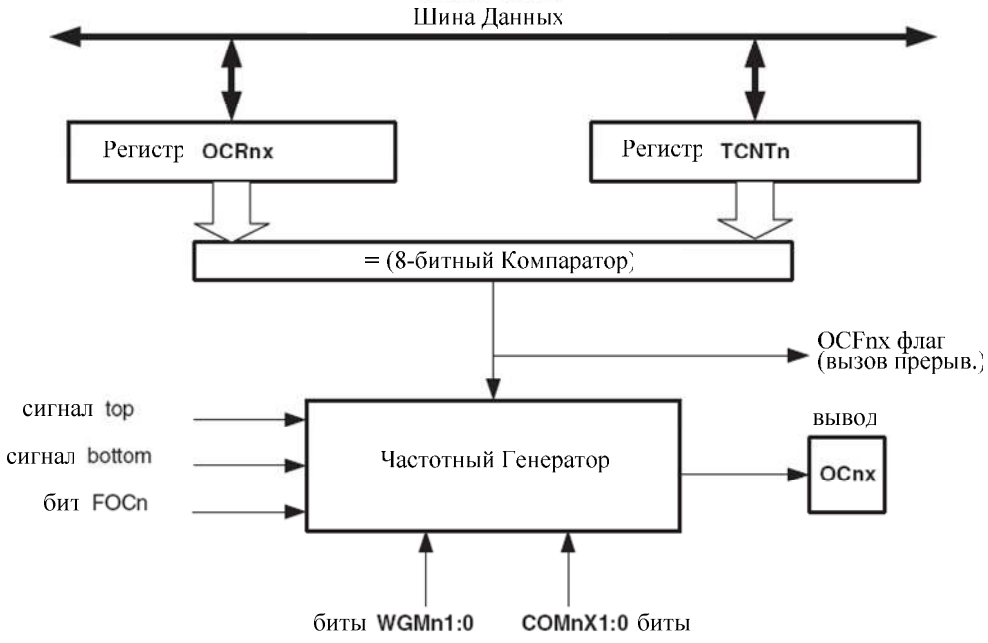



Рисунок 28 - Блок-диаграмма Модуля Сравнения

При использовании любого режима ШИМ (Широтно-Импульсной Модуляции) Регистры Сравнения OCR0x будут Дважды Буферизированными. Для Нормального Режима и Режима Очищения Таймера по Совпадению возможность использовать Двойную Буферизацию Регистров Сравнения недоступна. Двойная Буферизация синхронизирует обновление Регистров OCR0x со значениями TOP или BOTTOM!. Такая синхронизация предотвращает появление импульсов ШИМ нечетной длительности и несимметричных импульсов ШИМ, что, в свою очередь, позволяет добиться импульсов ШИМ свободных от дрожания (в общем, - избавиться от сбоев).

Работа с Регистром OCR0x может показаться сложной задачей, но это не так. Если Двойная Буферизация Разрешена, то ЦПУ обращается к Буферизированному Регистру OCR0x, а, если Двойная Буферизация Недоступна, то ЦПУ обращается к Регистру OCR0x напрямую.

# Форсированный Режим Сравнения

Если Генератор Частоты работает в режимах non-ШИМ (это Режимы, не являющиеся ШИМ-режимами), то Сравнение Регистров может быт форсировано, для этого надо записать единицу в бит FOC0x. При совпадении Сравниваемых значений флаг OCF0x устанавливаться не будет, также, не будет происходить Перезагрузка/Сброс таймера, но будет происходить обновление значения на выводе OC0x (настройка битов COM0x1:0 определяет, что будет происходить на выводе OC0x, очищение или переключение бита).

# Блокирование Режима Сравнения при Записи Регистра TCNT0

Любая операция записи в Регистр TCNT0 будет блокировать Режим Сравнения на время одного тактового импульса Таймера $clk_{T0}$ даже, если T/C0 остановлен. Это позволяет загружать в Регистр Сравнения OCR0x такое же значение, как и в Регистре Счетчика TCNT0 и после этого, если T/C0 включен, Прерывание по Совпадению вызывано не будет.

# Использование Модуля Сравнения

Так как запись в TCNT0 в любом режиме работы блокирует Режим Сравнения на один следующий тактовый цикл T/C0, то мы рискуем совершить ошибку при изменении значения в TCNT0, когда Регистр TCNT0 используется Модулем Сравнения, независимо от того, запущен T/C0 или - нет. Если значение, записываемое в TCNT0 будет эквивалентно значению в Регистре OCR0x, то на следующем после записи тактовом цикле Таймера Режим Сравнения Будет пропущен, что в результате приведет к некорректной работе Генератора Частоты. Точно так же, нельзя записывать в TCNT0 значение, равное BOTTOM, если T/C0 вычисляет Декремент.

Установка значения в вывод OC0x должна быть выполнена до настройки вывода в качестве Выхода, путем настройки Регистра DDR. Самый простой способ установить значение OC0x - это использовать в Нормальном Режиме стробирующий бит FOC0x. Регистр OC0x сохраняет свои значения даже при смене режима работы Генератора Частоты. Помните, что биты COM0x1:0 не являются буферизированными так же, как Сравниваемые Значения Регистров Сравнения. Изменение битов COM0x1:0 дают немедленный эффект.

# Режим Сравнения

Биты Режима Сравнения COM0x1:0 имеют две функции. Генератор Частоты использует биты COM0x1:0 для определения состояния на выводе OC0x при следующем Совпадении значений Регистров Сравнения. Также, биты COM0x1:0 управляют источником значений на Выходе OC0x. На Рисунке 29 показана упрощенная блок-схема работы битов . I/O-регистры, I/O-биты и I/O-выводы на рисунке подписаны жирными буквами. На Рисунке 29 показана только часть общих I/O-Регистров Управления (DDR и PORT), на работу которых влияют, биты COM0x1:0. При обращении к состоянию OC0x, идет обращение к Регистру OC0x, а не к Выводу OC0x. Если произойдет Сброс Системы, то Регистр OC0x тоже будет сброшен в "0".

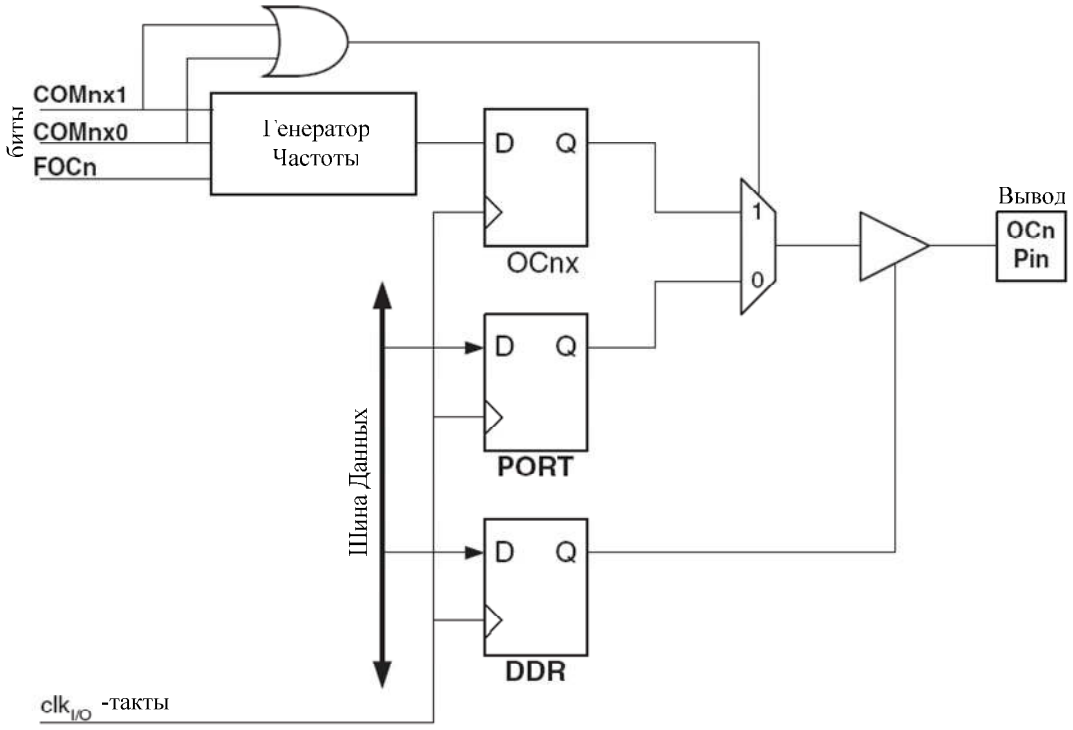



Рисунок 29 – Часть общих I/O-Регистров Управления (DDR и PORT), на работу которых влияют, биты COM0x1:0

Стандартная функция ввода/вывода будет проигнорирована Режимом Сравнения Генератора Частоты, если хоть один бит из COM0x1:0 битов будет установлен. Однако, направление в/в данных через Ножку OC0x все еще контролируется Регистром DDR. Бит в Регистре DDR (DDR_OC0x), настраивающий вывод OC0x как Выход, должен быть установлен до того, как Ножка OC0x будет использована в качестве Выхода. Функция Игнорирования других функций не зависит от того, в каком режиме работает Генератор Частоты.

Схема Модуля Сравнения позволяет инициализировать Регистр OC0x до того, как вывод OC0x будет использован в качестве Выхода. Обратите внимание, что настройка битов COM0x1:0 остается неизменной, что обеспечивает устойчивую работу во всех режимах. Обратитесь к разделу "Описание 8-битного Таймера/Счетчика 0" на странице 72 за подробной информацией.

# Модуль Сравнения и Генератор Частоты

Генератор Частоты в различных Режимах Работы (Normal, CTC, PWM(ШИМ)) использует биты COM0x1:0 по-разному. Для всех Режимов такая установка битов COM0x1:0=0 (подробнее: COM0x1=0 и COM0x0=0) говорит о том, что Генератор Частоты не пользуется Регистром OC0x. То, как работает Режим Сравнения в Режиме Таймера non-ШИМ (не ШИМ), можно посмотреть на Рисунке 28, страница 63. Режим Быстрый-ШИМ (fast-PWM) описан в Таблице 26 на странице 54, а Фазовая Коррекция ШИМ описана в Таблице 27 на странице 54.

/
/
/

# Режимы Работы

Режим работы, т.е поведение Таймера/Счетчика и Модуля Сравнения, определяется комбинацией битов WGM2:0 и COMx1:0. Биты COMx1:0 никак не влияют на вычисление Инкремента или Декремента Таймера/Счетчика, в отличии от битов WGM2:0. Биты COMx1:0 влияют на Генерирование импульсов ШИМ, и определяют - будут ли они Инверсными или - нет (Инверсный ШИМ или НЕИнверсный ШИМ). Для Режима non-ШИМ биты COMx1:0 определяют, будет ли в Режиме Сравнения бит на Выходе: 1-Установлен; 2-Очищен; 3-Переключен (обращайтесь к разделу "Модуль Сравнения" на странице 64).

За подробной информацией от Таймингах (Временных Задержках), обратитесь к Рисункам: 33, 34, 35 и 36 в разделе "Временные Диаграммы Таймера/Счетчика" на странице 70.

# Режим Normal

Нормальный Режим ("Normal") работы является наиболее простым из всех (биты устанавливаются так:WGM02:0 = 0). В Этом Режиме направление счета всегда является Инкрементирующим, а Очищение счетчика не выполняется. Счетчик просто переполняется, когда достигает максимального 8-битного значения (TOP=0xFF), а затем перезапускается заново со значения (0x00. В Normal Режиме Флаг Переполнения T/C0 -TOV0- будет устанавливаться на том же цикле, на котором будет происходить обнуление Регистра TCNT0. В этом случае Флаг TOV0 ведет себя так, как будто он является девятым битом Регистра TCNT0 (хотя, это и не так), за исключением того, что этот Флаг устанавливается, а не сбрасывается. Однако, в комбинации с Прерыванием по Переполнению, которое автоматически очищает Флаг TOV0, Разрешение (диапазон) счетчика T/C0 может быть Увеличено программно. Новое значение в Режиме Normal может быть записано в TCNT0 в любое время.

Модуль Сравнения может быть использован для вызова прерывания в нужное время. Не рекомендуется использовать Режим Сравнения для того, чтобы Генерировать какую-то частоту, так как это займет значительное время у ЦПУ.

# CTC-режим (Очищение Таймера при Совпадении Регистров Совпадения)

Аббревиатура "CTC" расшифровывается как "**Clear Timer on Compare**", что по-русски означает "Очищение Таймера при Совпадении значений Регистров Совпадения". В CTC-режиме (когда биты установлены так:WGM02:0 = 2) Регистр OCR0A используется для того, чтобы управлять/настраивать Разрешение Счетчика. В CTC-режиме значение Регистра Счетчика TCNT0 сбрасывается в нули, если значение TCNT0 соответствует значению, записанному в Регистр Сравнения OCR0A. Для Счетчика значение Регистра OCR0A будет являться TOP-значением, соответственно, таким образом можно настраивать Разрешение T/C0. Такой Режим предоставляет много возможностей для управления Генерированием частоты, используя Модуль Сравнения.

Временные диаграммы для CTC-режима показаны на Рисунке 30. Значение Счетчика в Регистре TCNT0 увеличивается до тех пор, пока число в TCNT0 не станет равным числу в Регистре OCR0A, после этого Счетчик TCNT0 будет очищен.

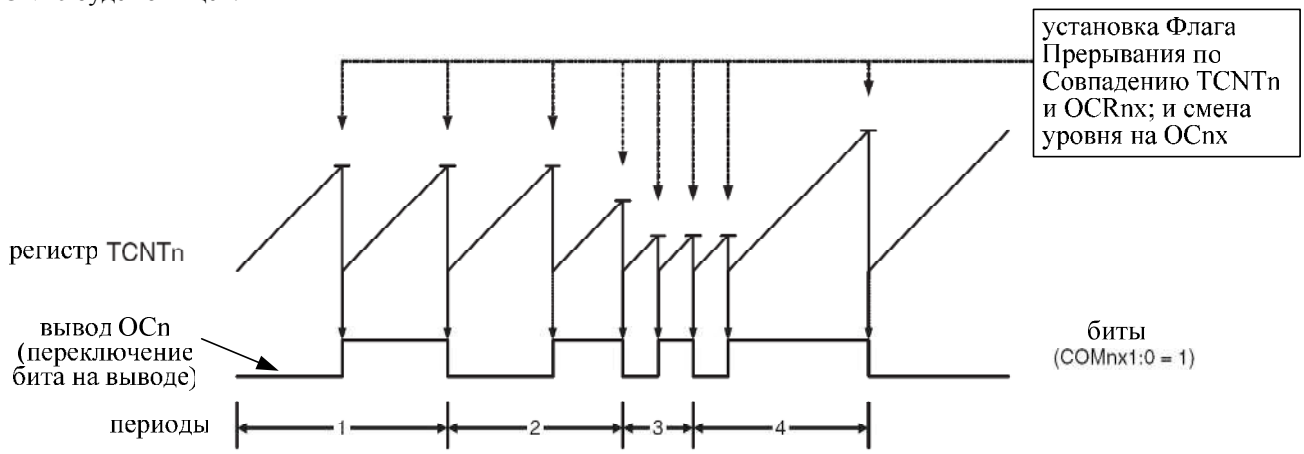



Рисунок 30 - Временные диаграммы для CTC-режима

Прерывание может Вызываться каждый раз, когда Регистр TCNT0 будет достигать значения TOP и будет устанавливаться Флаг OCF0A. Если Прерывания разрешены, то Подпрограмма Обработки Прерывания может быть использована для того, чтобы обновлять TOP-значение. Однако, следует помнить, что, если значение TOP близко к значению BOTTOM, то записывать его в Регистр следует очень осторожно, так как CTC-режим не работает с Дважды Буферизированными Регистрами, поэтому нужно учитывать частоту тактирования T/C0. Если в Регистр Сравнения OCR0A записать значение меньшее, чем значение уже присутствующее в Регистре Счетчика TCNT0, то Таймер/Счетчик 0 пропустит Сравнение Значений этих Регистров. В этом случае T/C0 должен будет досчитать до максимального значения 8-битного Регистра TCNT0 (0xFF), а за тем, начать счет с нуля (0x00) и считать до тех пор, пока значение в Регистре TCNT0 не станет равным значению в OCR0A.

Для Генерирования Частоты в CTC-режиме, следует вывод OC0A настроить на Режим Переключения своего логического уровня каждый раз, когда будут равны Регистры TCNT0 и OCR0A, это делается путем установки соответствующих битов: (COM0A1:0 = 1). Значение на выводе OC0A не будет изменяться, если этот вывод будет настроен как Выход. Максимальная частота, вырабатываемая таким образом, может достигать

значения в Два раза меньшего, чем Тактовая (Системная) Частота МК: $f_{OC0} = f_{clk\_I/O}/2$, при условии, что в Регистр OCR0A загружены нули (0x00). Частота Генератора в CTC-режиме определяется следующим Уравнением:

$$f_{OCnx} = \frac{[f_{clkI/O}]}{2 \cdot N \cdot (1 + OCRnx)}$$

Переменная N представляет собой коэффициент деления Тактовой Частоты МК, которая, в последствии, является Тактовой Частотой T/C0, эта переменная может принимать значения такого ряда: (N=1, N=8, N=64, N=256 или N=1024). Как и в Нормальном Режиме (Normal) работы T/C0, в Этом Режиме (CTC-режиме) Флаг TOV0 устанавливается на том же самом Тактовом Цикле, на котором происходит переключение Значения Регистра TCNT0 с максимального значения (MAX) на начальное нулевое значение (0x00).

# Режим Быстрой ШИМ-модуляции (FastPWM или FPWM-режим)

Режим Быстрой ШИМ-модуляции, или FPWM-режим (WGM02:0 = 3 или 7), позволяет Генерировать высокочастотный сигнал с ШИМ. FPWM-режим _ использует Однонаправленный Счет. Т.е. Счетчик считает от BOTTOM-значения до TOP-значения, затем Счетчик начинает считать заново от BOTTOM до TOP и т.д.. Значение TOP равно числу 0xFF, когда биты WGM2:0 = 3 или же TOP может быть равно числу, записанному в Регистр OCR0A, когда биты WGM2:0 = 7. Если используется НеИнвертирующий Режим Сравнения, то значение на выводе OC0x очищается (устанавливается ноль) при совпадении Регистров TCNT0 и OCR0x, а при достижении Регистром TCNT0 значения BOTTOM, на выводе OC0x обратно устанавливается единица. А, если используется Инвертирующий Режим Сравнения, то значение на выводе OC0x устанавливается (выгружается единица) при совпадении Регистров TCNT0 и OCR0x, а при достижении Регистром TCNT0 значения BOTTOM, на выводе OC0x обратно устанавливается ноль. Модуль PWM в FPWM-режиме, используя Однонаправленный Счет, позволяет Генерировать удвоенную частоту, такую же, какую может Генерировать ШИМ с Фазовой Коррекцией, используя Двунаправленный Счет. Способность Генерировать такую высокую частоту делает FPWM-режим отлично приспособленным для использования в приложениях Регулирования Мощности, _ в ЦАП-приложениях и др.. Использование в приложениях ВЧ-сигналов дает возможность применять радиоэлементы для поверхностного монтажа (катушки, конденсаторы и др.), что, в свою очередь, ведет к уменьшению стоимости проекта.

В FPWM-режиме Счетчик увеличивается до тех пор, пока Регистр TCNT0 не сравняется со значением TOP. Когда значение в TCNT0 достигнет значения TOP, произойдет очистка Регистра Счетчика TCNT0 сразу же на его следующем Тактовом Цикле. Временные Диаграммы для FPWM-режима показаны на Рисунке 29. Нарастающие значения Регистра TCNT0 на Рисунке 29 показаны как гистограмма (пилообразные импульсы), что иллюстрирует Однонаправленный Счет (импульс нарастает - значение в TCNT0 увеличивается). На Рисунке 29, также, показаны Инвертирующий и НеИнвертирующий режимы ШИМ. Короткие горизонтальные линии (Рисунок 29) на TCNT0-значениях представляют собой случай совпадения значений регистров OCR0x и TCNT0.

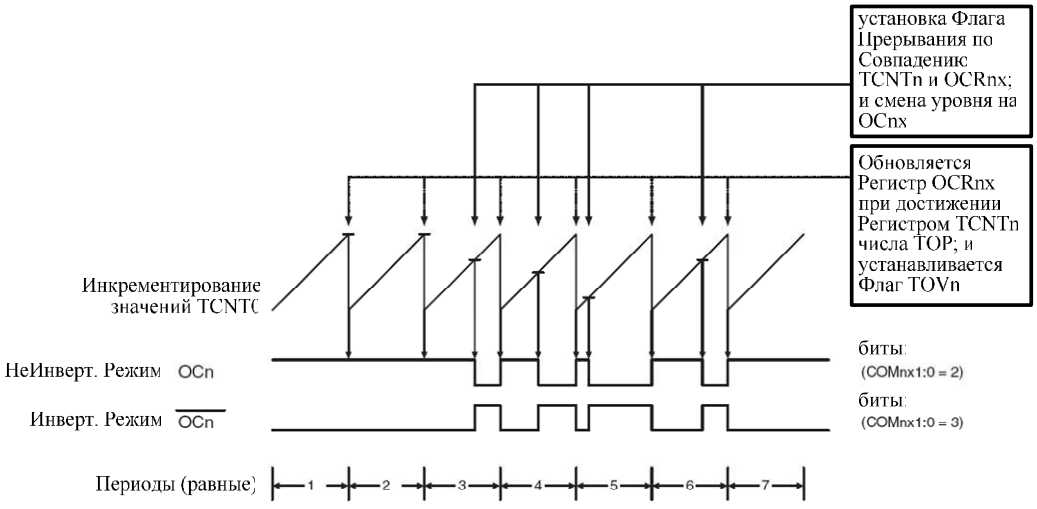



Рисунок 31 - Временные Диаграммы для FPWM-режима

Флаг Переполнения Таймера/Счетчика "TOV0" устанавливается каждый раз, когда Счетчик достигает значения TOP. Если прерывания разрешены, то можно использовать подпрограмму обработки прерываний для

того, чтобы обновлять (изменять) сравниваемые значения в Регистрах Сравнения. В FWPM-режиме Модуль Сравнения позволяет Генерировать ВЧ ШИМ-сигнал и выводить его на Вывод OC0x. Если настроить биты COM0x так: COM0x1:0 = 2, то это приведет ко включению НеИнвертирующего Режима ШИМ. Если настроить биты COM0x так: COM0x1:0 = 3, то это приведет ко включению Инвертирующего Режима ШИМ: если настроить биты COM0x так: COM0A1:0 = 1, то это позволит задействовать Вывод AC0A и переключать значение на нем в Режиме Сравнения, при том условии, что установлен бит WGM02. Эта Опция не доступна для использования Выводом OC0B (Смотрите Таблицу 54). Значение на этом Выводе (в этом случае) будет изменяться, если этот Вывод настроен как Выход. Принцип Генерирования ШИМ Частоты состоит в том, что в Режиме Сравнения на Выводе OC0x постоянно изменяется логический уровень сигнала, т.е. происходит установка в "1" (или очистка в "0"); или наоборот - очистка в "0" (или установка в "1") одновременно на том же Тактовом Цикле, когда Счетчик сбрасывает свое значение с TOP до BOTTOM.

Частота ШИМ, "снимаемая" с вывода OCnx, может быть рассчитана по Формуле:

$$f_{OCnx}PWM = \frac{[f_{clkI/O}]}{N \cdot 256}$$

Переменная N представляет собой коэффициент деления Тактовой Частоты МК, которая, в последствии, является Тактовой Частотой T/C, эта переменная может принимать значения такого ряда: (N=1, N=8, N=64, N=256 или N=1024).

Критические Значения для Регистра OCR0A представляют собой особый случай при Генерировании ШИМ-частоты. Если значение Регистра OCR0A эквивалентно значению BOTTOM, то на Выходе будет происходить выброс импульса на каждом следующем Тактовом Цикле Таймера после достижения числа MAX: MAX+1; Загрузка в OCR0A числа, равного значению MAX, в результате дает постоянный логический уровень на Выходе - высокий или низкий (последнее зависит от полярности Выхода, которая зависит от установки битов COM0A1:0).

ШИМ-частота в FPWM-режиме с Рабочим Циклом 50% (когда длительность импульса равна половине периода цикла) может быть получена переключением логического уровня на Выводе OC0x используя Режим Сравнения (COM0x1:0 = 1). Максимальная частота, вырабатываемая таким образом, может достигать значения в Два раза меньшего, чем Тактовая (Системная) Частота МК: $f_{OC0} = f_{clk\_I/O}/2$, при условии, что в Регистр OCR0A загружены нули (0x00). Этот способ Генерирования Частоты подобен CTC-режиму, в котором используется переключение вывода OC0A, за исключением того, что в FPWM-режиме Двойная Буферизация Модуля Сравнения Разрешена, а в CTC-режиме - Запрещена.

# ШИМ с Фазовой Коррекцией (PCPWM-режим)

PCPWM-режим (WGM02:0 = 1 или 5) позволяет воспользоваться высоким Разрешением Генератора Частоты и обеспечивает ШИМ-сигнал правильный по фазе. PCPWM-режим основан на Двунаправленном Счете. При Двунаправленном Счете Счетчик циклически считает от значения BOTTOM до TOP,а затем, обратно от значения TOP до BOTTOM. Значение TOP будет равно 0xFF, если WGM2:0 = 1, а, если установлены биты WGM2:0 = 5, то значение TOP будет равно числу, загруженному в OCR0A. При использовании НеИнвертирующего Режима Сравнения и Инкрементированного Счета, значение на Выводе OC0x будет Очищаться в "0" при совпадении Регистров TCNT0 и OCR0x и наоборот, при том же Режиме Сравнения, значение на Выводе OC0x будет Устанавливаться в "1" при совпадении TCNT0 и OCR0x при Декрементированном Счете. При использовании Инвертирующего Режима Сравнения все работает наоборот. Двунаправленный Режим Счета имеет более низкую максимальную частоту генерирования, чем Однонаправленный. Однако, из-за своей симметричности Двунаправленный Режим Счета предпочтительно использовать для приложений, управляющих двигателями.

В PCPWM-режиме Счетчик увеличивается (Инкрементируется) до тех пор, пока Регистр TCNT0 не достигнет значения TOP. При достижении значения TOP Счетчик меняет направление Счета. Естественно, что TCNT0 равен TOP только на время одного Тактового Цикла. Временные диаграммы для PCPWM-режима показаны на Рисунке 32. Увеличивающееся и уменьшающееся значения Регистра TCNT0 показаны на Рисунке 32 как гистограммы (кривые), они иллюстрируют Двунаправленный Режим Счета. На Рисунке 32, также, показаны временные диаграммы значений на Выходах OCn для Инвертирующих и НеИнвертирующих Режимов Сравнения. Короткие горизонтальные линии на гистограммах представляют собой время совпадения между Регистрами OCR0x и TCNT0.

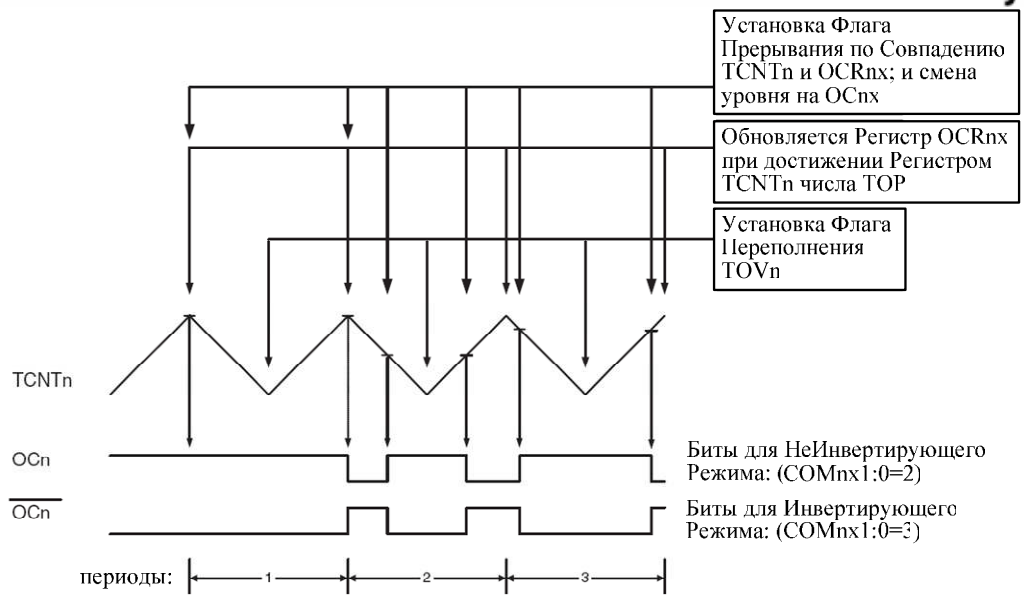



Рисунок 32 - Временные диаграммы значений на Выходах OCn для Инвертирующих и НеИнвертирующих Режимов Сравнения (PCPWM-режим)

Флаг Переполнения (TOV0) Таймера/Счетчика устанавливается каждый раз, когда Счетчик достигает значения BOTTOM. Этот Флаг можно использовать для того, чтобы вызывать прерывания.

В PCPWM-режиме Модуль Сравнения позволяет Генерировать ШИМ-частоту и, затем, выводить ее на Вывод OC0x. Если настроить биты так: COM0x1:0 = 2, то Модуль Сравнения будет работать в НеИнвертирующем Режиме. Инвертирующий Режим включается, если настроить биты так: COM0x1:0 = 3, загрузка "единицы" в бит COM0A0 позволяет переключать значение на Выводе OC0A каждый раз, когда Регистры Сравнения равны между собой, при условии, что установлен бит WGM02. Вывод OC0B настраивается по-другому. Значение на Выводе OC0x будет видимо, если этот Вывод настроен как Выход. ШИМ-частота генерируется на Выводе OC0x путем очищения в "0" (или установки в "1" для Инвертирующего Режима) логического уровня на этом Выводе, когда Регистры Сравнения OCR0x и TCNT0 (точнее OCR0x - это Регистр Сравнения, а TCNT0 - это Регистр Счетчика, значение которого сравнивается со значением OCR0x) становятся равны между собой при Инкрементированном Счете; и путем установки в "1" (или очищением в "0" для Инвертирующего Режима), когда Регистры Сравнения OCR0x и TCNT0 становятся равны между собой при Декрементированном Счете. Генерируемая Частота в PCPWM-режиме может быть рассчитана по следующей Формуле:

$$f_{OCnx}\,PCPWM = \frac{[f_{clkI/O}]}{N \cdot 510}$$

Переменная N представляет собой коэффициент деления Тактовой Частоты МК, которая, в последствии, является Тактовой Частотой Т/С, эта переменная может принимать значения такого ряда: (N=1, N=8, N=64, N=256 или N=1024).

Критические Значения (т.е., создающие риск ошибки в работе) для Регистра OCR0A представляют собой особый случай при Генерировании ШИМ-частоты в PCPWM-режиме. Если OCR0A эквивалентен значению BOTTOM, то на Выводе всегда будет низкий уровень сигнала, а, если OCR0A=TOP, то на Выводе всегда будет высокий уровень сигнала - это справедливо для НеИнвертирующего Режима Сравнения. Для Инвертирующего Режима все будет наоборот.

В самом начале 2-го периода, Рисунок 32, значение на Выводе OCn меняет свое значение с Высокого Уровня на Низкий даже, если Регистры Сравнения не равны. Это гарантирует генерирование симметричных импульсов в Двунаправленном Режиме Счета. Это второй случай, когда происходит смена логического уровня без Сравнения Регистров Сравнения.

• Регистр OCR0A может иметь значения не равные MAX, как показано на Рисунке 32. Если OCR0A=_TOP(MAX)_, то значение на Выводе OCn будет таким же, какое получается в результате равенства Регистров Сравнения при Декрементированном Счете, а это "1" - при НеИнвертирующем Режиме и "0"- при Инвертирующем Режиме.

Симметрия Импульсов ШИМ-частоты гарантирована, если значение на Выводе OCn, при TCNTn=MAX, соответствует значению, получающемуся в результате равенства Регистров Сравнения при Инкрементирующем Счете.

• Таймер начинает считать с числа на единицу большего, чем то, которое загружено в OCR0A (т.е. OCR0A+1) и поэтому, пропускает Сравнение на один период (даже, если TCNT0 и OCR0A равны) и, следовательно, не происходит ошибочного изменения сигнала на выводе OCn (что показано на Рисунке 32 – период 1 там пропущен).

# Временные Диаграммы Таймера/Счетчика

T/Cn является Синхронным Модулем, а Тактовые Импульсы $clk_{TO}$, как показано на следующем рисунке, являются Синхронизирующими. На Рисунке, также, видно, что происходит после установки Флага Прерывания. Рисунок 33 иллюстрирует, как Данные Счетчика изменяются во времени. На Рисунке видно, что Счетчик считает до Значения MAX, это относится ко всем ШИМ-режимам, кроме PCPWM-режима..

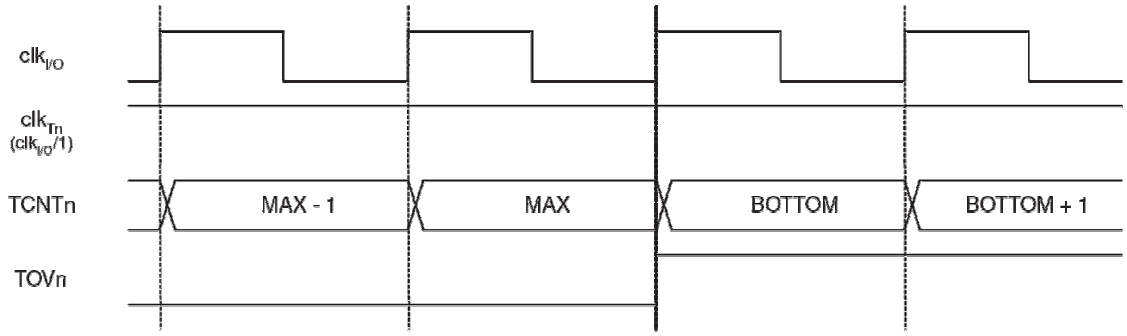



**Рисунок 33** - Временные Диаграммы Таймера/Счетчика (Предделитель не используется)

Рисунок 34 показывает те же процессы, что на Рисунке 33, но с использованием Предделителя ($f_{clkI/O}/8$)

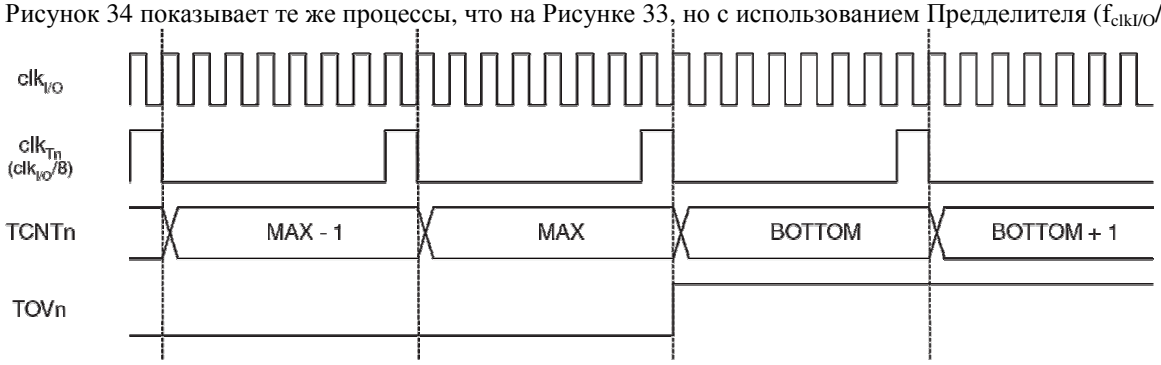



**Рисунок 34** Временные Диаграммы Таймера/Счетчика при использовании Предделителя ($f_{clkI/O}/8$)

Рисунок 35 иллюстрирует установку Флагов Сравнения OCF0B и OCF0A (в Режиме Сравнения Регистров Сравнения) во всех ШИМ-режимах, за исключением CTC-режима и ШИМ-режима, где Регистр Сравнения OCR0A является TOP-значением.

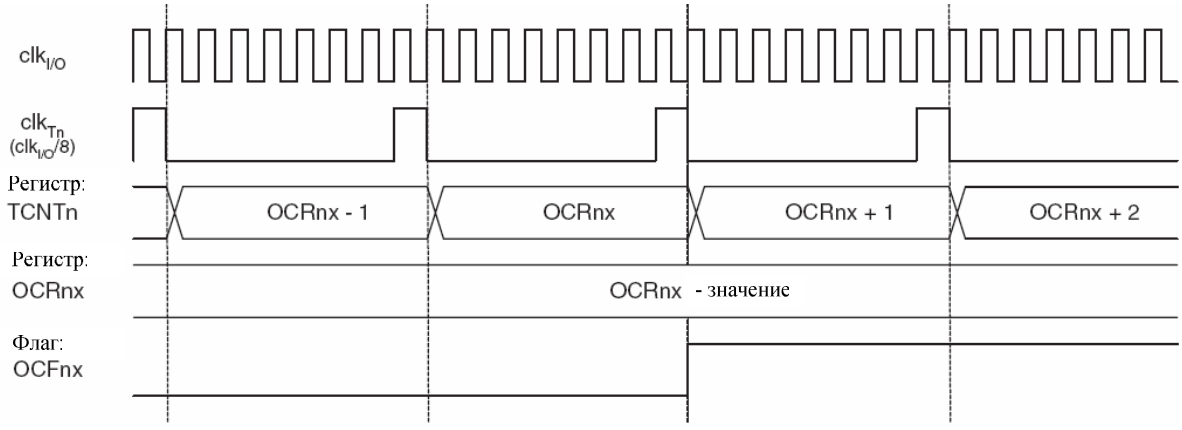



**Рисунок 35** - Временные Диаграммы Таймера/Счетчика с Установкой Флагов OCF0x с использованием Предделителя ($f_{clkI/O}/8$)

Рисунок 36 показывает установку Флага OCF0A и очищение TCNT0 в CTC-режиме и FPWM-режиме, где OCR0A является TOP-значением.

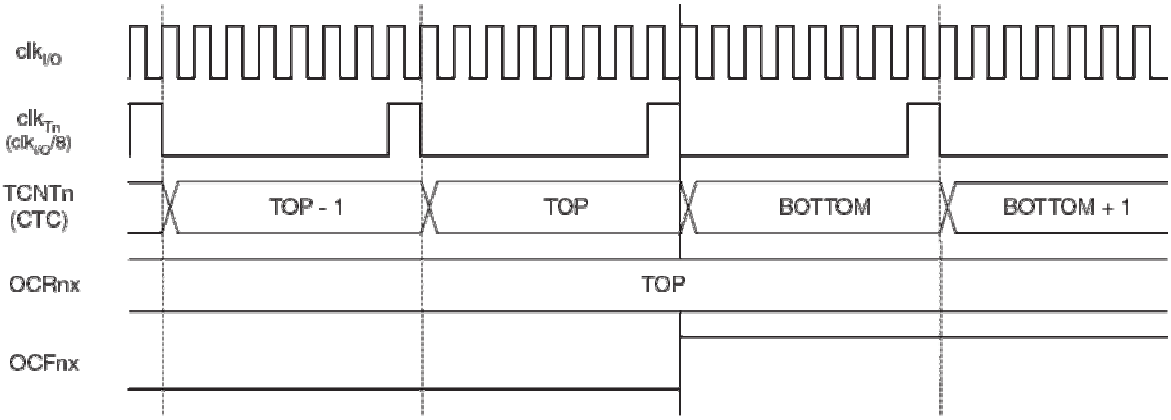



**Рисунок 36** - Временные Диаграммы Таймера/Счетчика с Очисткой Таймера в Режиме Сравнения с использованием Предделителя ($f_{clkI/O}/8$)

# Описание Регистров 8-битного Таймера/Счетчика 0

## Регистр Управления А Таймера/Счетчика 0 – «TCCR0A»

| Бит № | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | COM0A1 | COM0A2 | COM0B1 | COM0B0 | --- | --- | WGM01 | WGM00 | TCCR0A |
| Чтение/ Запись | Ч/З | Ч/З | Ч/З | Ч/З | Ч | Ч | Ч/З | Ч/З | |
| Начальные Значения | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**• Биты 7:6 - COM0A1:0:Включен Режим Сравнения А**

Эти биты управляют Выводом OC0A. Если из битов COM0A1:0 установлен хотя бы один (или два бита), то Вывод OC0A будет игнорировать Нормальные Функции Порта в/в. Однако, обратите внимание, что бит, соответствующий в Регистре DDR этому Выводу, должен быть установлен так, чтобы был включен Драйвер Вывода.

Если Альтернативная Функция OC0A «подключается» к своему одноименному Выводу, то Функция, включаемая битами COM0A1:0, становится зависимой от настроек битов WGM02:0. В Таблице 34 показано, какую функцию несут биты COM0A1:0 при установленных WGM02:0 битах в Нормальном Режиме (Normal-режим) или CTC-режиме (в non-PWM режимах - не являющимися ШИМ-режимами Таймера/Счетчика).

Таблица 34 – Режим Сравнения в non-PWM (non-ШИМ) Режимах Таймера

| COM0A1 | COMA0 | Описание |
|---|---|---|
| 0 | 0 | Нормальная работа порта в/в, Функция (Вывод) OC0A отключена |
| 0 | 1 | Переключение (инвертирование) значения на Выводе OC0A в Режиме Совпадения Регистров Сравнения |
| 1 | 0 | Очищение (в «0») значения на Выводе OC0A в Режиме Совпадения Регистров Сравнения |
| 1 | 1 | Установка (в «1») значения на Выводе OC0A в Режиме Совпадения Регистров Сравнения |

Таблица 35 – Режим Сравнения Регистров в FPWM-режиме Таймера

| COM0A1 | COMA0 | Описание |
|---|---|---|
| 0 | 0 | Нормальная работа порта в/в, Функция (Вывод) OC0A отключена |
| 0 | 1 | WGM02 = 0: Вывод OC0A отключен (Ножка PB2 не является Выводом OC0A, но является Выводом PB2, выполняющим обычные функции порта). WGM02 = 1: Переключение (инвертирование) значения на Выводе OC0A в Режиме Совпадения Регистров Сравнения. |
| 1 | 0 | Очищение (в «0») значения на Выводе OC0A в Режиме Совпадения Регистров Сравнения, Установка происходит при достижении TCNT0 значения TOP. |
| 1 | 1 | Установка (в «1») значения на Выводе OC0A в Режиме Совпадения Регистров Сравнения, Установка происходит при достижении TCNT0 значения TOP. |

Примечание:

1. Особый Случай представляет собой случай, когда Регистр OCR0A равен значению TOP при установленном бите COM0A1. В этом Случае Режим Сравнения игнорируется (не работает), а на Выводе OC0A присутствует постоянный логический уровень - высокий или низкий (последнее зависит от полярности Выхода, которая зависит от установки битов COM0A1:0). Обращайтесь к Разделу Режим «Режим Быстрой ШИМ-модуляции» на странице 67 (см. предпоследний абзац).

Таблица 36 демонстрирует работу битов COM0A 1:0, когда через биты WGM0 2:0 включен режим Фазовой Коррекции ШИМ – PCPWM.

Таблица 36 – Режим Сравнения Регистров при включенном режиме Фазовой Коррекции ШИМ – PCPWM.(1).

| COM0A1 | COMA0 | Описание |
|---|---|---|
| 0 | 0 | Нормальная работа порта, OC0A отключен. |
| 0 | 1 | WGM02 = 0: Нормальная работа порта, OC0A отключен.<br>WGM02 = 1: Переключение уровня на OC0A при совпадении регистров в Режиме Сравнения. |
| 1 | 0 | OC0A очищается по Совпадению Регистров при инкрементном счете. И OC0A устанавливается в 1 по Совпадению Регистров при декрементном счете Таймера. |
| 1 | 1 | |

Примечание:

1. Особый случай бывает, когда Регистр OCR0A эквивалентен значению TOP и установлен бит COM0A1. В этом случае Режим Сравнения Игнорируется. Подробнее об этом написано в разделе «ШИМ с Фазовой Коррекцией (PCPWM-режим)» на стр. 68.

# **USART –** Универсальный Синхронный и Асинхронный Последовательный Приемник и Передатчик

1. Полнодуплексный режим работы (Независимые Регистры для Передатчика и Приемника с последовательной загрузкой данных);
2. Асинхронный и Синхронный режим работы;
3. Master- или Slave-тактирование при Синхронном режиме работы;
4. Генератор Битрейта (Бодрейта) с высоким разрешением для настройки скорости передачи данных;
5. Поддерживаемый размер Кадра: 5- , 6- , 7- , 8- или 9-битный Кадр и 1 или 2 Стоповых Бита;
6. Генератор Равенства Четных и Нечетных битов, аппаратная проверка на равенство (чет-нечет);
7. Детектор переполнения Данных;
8. Детектор ошибки Кадра;
9. Фильтр Шума, определяющий истинность (ложность) Стартового Бита и Цифровой Низкочастотный Фильтр;
10. Три отдельных Прерывания, срабатывающие по окончанию Передачи Данных, при опустошении Регистра Данных при Передаче и по окончанию Чтения Данных Приемником;
11. Мультипроцессорный Режим Связи;
12. Двойная Скорость работы при Асинхронном Режиме Работы;

# Обзор

Упрощенная блок-диаграмма для USART – Передатчика приведена на рисунке 52. Обращение ЦПУ к Регистрам ввода/вывода и выводам МК на блок-диаграмме обозначено жирными линиями.

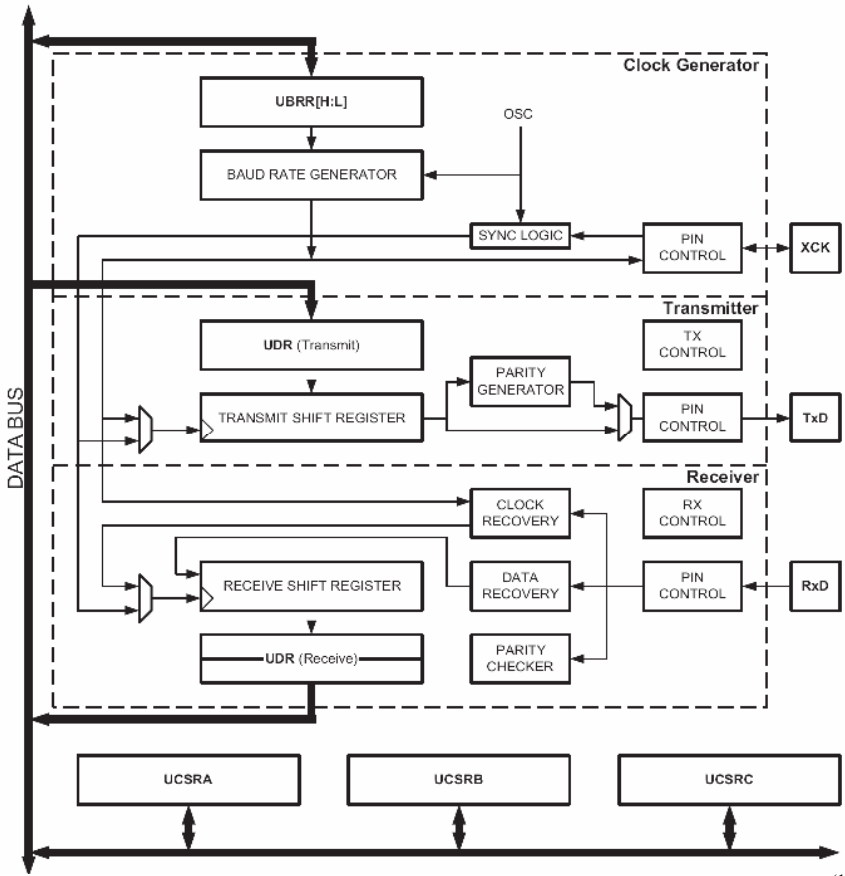



Рисунок 52 - Упрощенная блок-диаграмма для USART – Передатчика(1)

Примечание: 1. Чтобы уточнить информацию о расположении выводов интерфейса USART, обратитесь к Рисунку 1 на странице 2, -к Таблице 29 на странице 56, -к Таблице 26 на странице 54.

Пунктирными линиями на блок-диаграмме отделены три главные части модуля USART (расположенные сверху в низ): Тактовый Генератор, Передатчик и Приемник. Регистры Управления доступны всем модулям МК. Схема Тактового Генератора состоит из синхронизирующей логики для внешних Тактовых Сигналов, используемых в синхронном режиме работы и Бодрейт Генератора (the baud rate generator). Вывод XCK (Тактирование Передатчика) используется только при Синхронной работе Передатчика. Передатчик состоит из одного Буфера Данных, Последовательного Регистра Сдвига, Генератора Равенства и Логической Схемы Управления для обработки различных форматов Кадра. Буфер Данных позволяет непрерывно передавать данные без задержки между передаваемыми кадрами. Приемник является наиболее сложной составной частью модуля USART, что обусловлено Блоком Тактирования и Блоком Определения Стартового Бита. Блоком Определения Стартового Бита используется для асинхронного приема данных. Вдобавок к Блоку Определения Стартового Бита Приемник имеет Схему Проверки на Четность, Сдвиговый Регистр и Двухуровневый буфер (UDR) для приема данных. Приемник поддерживает те же форматы Кадра, что и Передатчик и может выявлять Ошибки Кадра, Переполнение Буферов и Ошибки на Четность.

# AVR USART «против» AVR UART – Совместимость

USART полностью совместим с модулем UART относительно следующего:

1. Расположение битов внутри Регистров USART;
2. Бодрейт Генератора;
3. Работы Передатчика;
4. Функционирования Буфера Передатчика;
5. Работы Приемника;

Однако, Буфер Приемника имеет два улучшения, которые влияют на совместимость в некоторых особых случаях:

1. Был добавлен Второй Буферный Регистр. Два Буферных Регистра работают, как замкнутый по кругу FIFO-буфер. Поэтому, Регистр Данных UDR следует читать только один раз по приходу данных. Более важными являются: возникновение Ошибки Кадра, о чем символизирует Флаг Ошибки (FE - «Frame Error» и DOR); и девятый бит данных (RXB8), приходящий вместе с полезными данными. Поэтому, прежде, чем осуществлять чтение Регистра UDR, следует читать Бит Статуса. В противном случае, информация об ошибке будет потеряна.
2. Сдвиговый Регистр Приемника теперь может являться Буфером третьего уровня. Этим можно пользоваться при приеме данных и «складывании» их в Сдвиговом Регистре (смотрите Рисунок 52), если Буфер Данных еще не был опустошен, когда появился новый Стартовый Бит. Таким образом, модуль USART более устойчив к переполнению Регистра Данных – (DOR - «Data OverRun»). Следующий Бит Управления теперь имеет новое имя, но, по-прежнему, выполняет первоначальные функции и имеет то же расположение в Регистре, что и раньше:
   - Бит CHR9 переименован на UCSZ2.
   - Бит OR переименован на DOR.

# Тактовый Генератор

Тактовый Генератор вырабатывает базовую частоту для работы Передатчика и Приемника. USART поддерживает 4 режима работы Тактового Генератора. Бит UMSEL в «Регистре Статуса и Управления – UCSRC» модуля USART осуществляет выбор между Асинхронным и Синхронным Режимом Работы. Двойная Скорость Работы (используется только в Асинхронном Режиме) выбирается битом U2X, который находится в Регистре UCSRA. При использовании Синхронного Режима (UMSEL=1), Регистр DDRx для вывода XCK (DDR_XCK) определяет какой источник тактирования будет использоваться, Внутренний или Внешний, соответственно, это Master-режим или Slave-режим.

На Рисунке 53 показана блок-диаграмма Тактового Генератора.

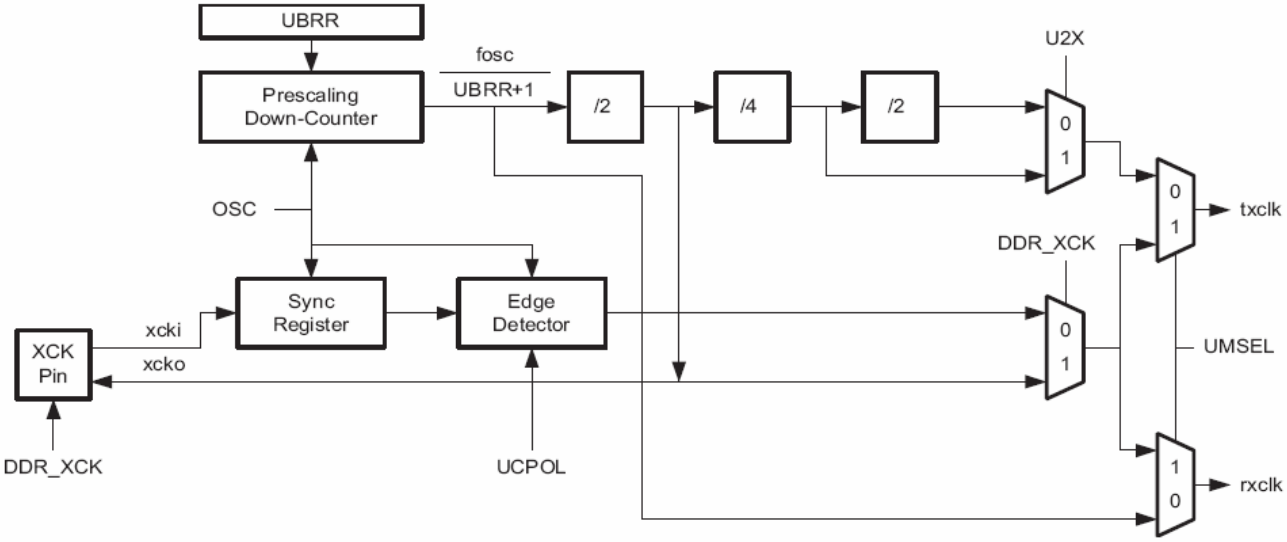



Рисунок 53 - Блок-диаграмма Тактового Генератора USART

Описание сигналов:

1. **txclk** – Тактовый Сигнал Передатчика (внутренний сигнал);
2. **rxclk** – Базовый Тактовый Сигнал Приемника (внутренний сигнал);
3. **xcki** – Вход с ножки XCK (внутренний сигнал). Используется для синхронизации в Режиме Slave.
4. **xcko** – Вывод Тактового Сигнала на ножкуXCK. Используется для синхронизации в Режиме Mastre.
5. **fosc** – XTAL-ножка (вывод для Системной Тактовой Частоты).

# Внутренний Тактовый Генератор - Бодрейт Генератор

Внутренний Тактовый Генератор используется для Асинхронной и Синхронной работы в режиме Master. Написанное в этом разделе относится к Рисунку 53. Регистр UBRR и Счетчик Обратного Счета выступают в качестве программируемого предделителя, получается Бодрейт Генератор. Счетчик Обратного Счета тактируется Системной Тактовой Частотой ($f_{osc}$), а настраивается он загрузкой значения из Регистра UBRR, которая происходит каждый раз при обнулении Счетчика Обратного Счета или при записи в Регистр UBRRL. Тактовый импульс вырабатывается каждый раз, когда происходит обнуление Счетчика Обратного Счета. Тактовая частота Бодрейт Генератора вычисляется по формуле: (output =$f_{osc}$/(UBRR+1)). Передатчик делит частоту Бодрейт Генератора output на 2, 8 или 16 в зависимости от Режима. Частота output используется непосредственно для тактирования Приемника и Блока Определения Стартового Бита Данных. _ .Таблица 48 содержит уравнения для расчета Бодрейта (Битрейта) и значения, которое следует загрузить в Регистр UBRR для каждого режима работы, использующего внутренний источник тактовой частоты.

Таблица 48 - Уравнения для вычисления Бодрейта и значения Регистра UBRR

| Режим работы | Уравнения для вычисления Бодрейта | Уравнения для вычисления значения Регистра UBRR |
|---|---|---|
| Асинхронный Нормальный Режим (U2X = 0) | $BAUD = \frac{f_{OSC}}{16(UBRR+1)}$ | $UBRR = \frac{f_{OSC}}{16BAUD} - 1$ |
| Асинхронный Режим с Удвоенной Скоростью(U2X = 1) | $BAUD = \frac{f_{OSC}}{8(UBRR+1)}$ | $UBRR = \frac{f_{OSC}}{8BAUD} - 1$ |
| Синхронный Master-режим | $BAUD = \frac{f_{OSC}}{2(UBRR+1)}$ | $UBRR = \frac{f_{OSC}}{2BAUD} - 1$ |

**BAUD** – Бодрейт - скорость передачи данных, представляет собой определенное количество передаваемых битов в секунду, чаще всего 1 бод=8 бит/с;

**$f_{OSC}$** – Системная тактовая частота;

**UBRR** – Это Регистр для настройки Бодрейта, состоит он из двух Регистров UBRRH и UBRRL, максимальное значение , которое можно записать в UBRR является 12-битным (0-4095). Несколько примеров значений для Регистра UBRR с соответствующими им частотами приведены в Таблице 56 (смотрите страницу 133).

# Двойная Скорость Работы (U2X)

Принимаемый поток данных может быть удвоен настройкой соответствующего бита U2X в Регистре UCSRA. Установка этого бита дает эффект только при Асинхронной работе USART. Этот бит следует сбрасывать в ноль, если используется Синхронный Режим Работы USART. Установка этого бита уменьшит коэффициент деления делителя для Бодрейта с 16 до 8, таким образом, повысив в два раза передачу данных при Асинхронном Режиме Работы USART. Однако, обратите внимание, что Приемник в этом случае использует только половину числа выборок (выборка уменьшится с 16 до 8 импульсов) для определения Битов Данных и определения Бита Синхронизации, и, таким образом, предъявляются повышенные требования к точности настройки Бодрейта и стабильности Системной Тактовой Частоты. _.

# Тактирование от Внешнего Источника

Тактирование от Внешнего Источника используется в Синхронном Slave-режиме работы. Написанное в этом разделе относится к Рисунку 53. Такты от Внешнего Источника поступают на ножку XCK в Регистр Синхронизации для обработки. Затем, эти такты с выхода Регистра Синхронизации должны пройти через Детектор Фронта и только после этого они могут быть использованы Передатчиком и Приемником. Этот процесс занимает два тактовых цикла ЦПУ, то есть, получается задержка, отсюда – максимальная частота Внешнего Источника, поступающая на ножку XCK, ограничена следующим уравнением:

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

Примечание: Значение $f_{osc}$ зависит от стабильности Системной Тактовой Частоты. Поэтому рекомендуется это учитывать для того, чтобы исключить возможную потерю данных.

# Работа в Синхронном Режиме

При Работе в Синхронном Режиме (UMSEL=1) вывод XCK используется как Тактовый Вход (Slave-режим) или как Тактовый Выход (Master-режим). Здесь принцип работы в том, что Входные Данные (поступающие на Вывод RxD) сэмплируются (осуществляется выборка) противоположным фронтом тактового импульса XCK Относительно того фронта, которым осуществляется передача данных на Вывод TxD:

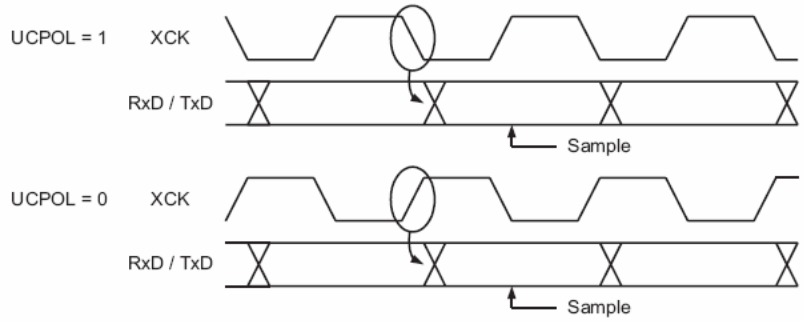



Рисунок 54 – Временные диаграммы для Синхронного режима

Биты UCPOL и UCRSC определяют, какой фронт сигнала на XCK используется для сэмплирования Входных Данных и Выходных. На Рисунке 54 показано, что, когда бит UCPOL = 0, то Биты Выходных Данных будут меняться по нарастающему фронту XCK, а сэмплироваться – по спадающему фронту XCK. Если бит UCPOL=1, то Биты Выходных Данных будут меняться по нарастающему фронту XCK, а Входные Данные сэмплируются по нарастающему фронту.

# Формат Кадра

Формат Кадра определяет свойства порции Битов Данных, где должны присутствовать биты синхронизации (стартовый и стоповый) и дополнительно можно внести Биты Четности. Модуль USART работает со всеми следующими комбинациями кадров:

- 1 - Стартовый Бит;
- 5,6,7,8 или 9 Битов Данных;
- Биты Четности (Нечетности) могут присутствовать или отсутствовать;
- 1 или 2 Стоповых Бита;

Кадр начинается со Стартового Бита, он является самым младшим. Следом за ним идут Биты Данных один за другим последовательно, заканчивается Кадр самым старшим битов. Если это разрешено, то Биты Четности (Нечетности) будут вставляться сразу после Битов Данных и, одновременно, перед Стоповым битом. Когда передача данного Кадра будет закончена, непосредственно за ним может последовать другой Кадр с новыми данными, в противном случае, линия связи переходит в Холостое Состояние, когда на линии присутствует высокий логический уровень. Рисунок 55 иллюстрирует возможные комбинации формата Кадра. Биты внутри скобок являются опционально включаемыми (если требуются).

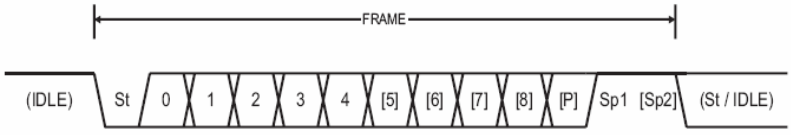



Рисунок 55 – Формат Кадра

- St – Стартовый Бит (всегда является низким лог. уровнем);
- (n) – Биты Данных (от 0 до 8);
- P – Бит Четности (Нечетности);
- FRAME – Весь Кадр;

- Sp – Стоповый Бит (всегда является высоким лог. уровнем);
- IDLE – Нет передачи на линии связи (RxD или TxD).

Формат Кадра для USART устанавливается Битами UCSZ2:0, UPM1:0 и USBS в Регистрах UCSRB и UCSRC. Приемник и Передатчик пользуются одинаковыми настройками. Обратите внимание, что смена настроек этих битов во время работы USART непременно дает сбой во время связи, как для Приемника, так и для Передатчика.

Биты SIZe (UCSZ2:0) настраивают число Битов Данных в Кадре. Биты UPM1:0 разрешают и настраивают какой тип Битов использовать Четности или Нечетности. Выбор между двумя или одним Стоповыми Битами осуществляет Бит USBS. Приемник игнорирует Второй Стоповый Бит. FE (Frame Error – Ошибка Кадра) будет определена только, если Первый Стоповый Бит равен нулю.

# Вычисление Бита Четности (Нечетности)

Вычисление идет Исключающим ИЛИ для всех Битов Данных. Если используется Бит Нечетности (Odd), то результат операции Исключающее ИЛИ инвертируется. Соотношения между Битами Четности (Нечетности) и Битами Данных следующие:

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

- $P_{even}$ – Бит Четности;
- $P_{odd}$ – Бит Нечетности;
- $d_n$ – Бит Данных;

Если Бит Четности (Нечетности) используется, то он располагается между последним Битом Данных и Первым Стоповым Битом.

# Инициализация USART (задание начальных условий)

Инициализация USART должна быть сделана прежде, чем этот модуль будет использован для осуществления связи. Процесс инициализации состоит из настройки Бодрейта, настройки формата Кадра и включения разрешения работы Передатчика и/или Приемника. Во время инициализации USART Прерывания должны быть запрещены Глобально. При реинициализацией, когда надо поменять Бодрейт или формат Кадра, надо удостовериться, что в этот момент не осуществляется обмен данными по линии связи. Флаг TXC может быть использован для того, чтобы определить, что передача данных была завершена. А Флаг RXC можно использовать для того, чтобы удостовериться, что в Буфере приемника не осталось не прочитанных данных. Обратите внимание, что Флаг TXC должен очищаться каждый раз перед передачей данных (перед записью данных в Регистр UDR), конечно, если этот Флаг используется в этих целях.

Следующие примеры кода для инициализации USART показывают одну Ассемблерную и одну С-функцию, которые между собой эквивалентны. Эти примеры предполагают Асинхронный Режим Работы USART с периодическим опросом флагов (когда прерывания запрещены) и фиксированный формат Кадра. Переменная для настройки Бодрейта является входным параметром функции. Для Ассемблерной функции переменная для настройки Бодрейта сохранена в Регистрах R17 и R16.

Assembly Code Example(1)

```
USART_Init:
  ; Set baud rate
  out  UBRRH, r17
  out  UBRRL, r16
  ; Enable receiver and transmitter
  ldi  r16, (1<<RXEN)|(1<<TXEN)
  out  UCSRB,r16
  ; Set frame format: 8data, 2stop bit
  ldi  r16, (1<<USBS)|(3<<UCSZ0)
  out  UCSRC,r16
  ret
```

C Code Example(1)

```
void USART_Init( unsigned int baud )
{
  /* Set baud rate */
  UBRRH = (unsigned char)(baud>>8);
  UBRRL = (unsigned char)baud;
  /* Enable receiver and transmitter */
  UCSRB = (1<<RXEN)|(1<<TXEN);
  /* Set frame format: 8data, 2stop bit */
  UCSRC = (1<<USBS)|(3<<UCSZ0);
}
```

Примечание: 1. Предполагается, что соответствующие заголовочные файлы подключены. _.

Более универсальные функции инициализации могут иметь в качестве входных параметров формат Кадра и запрещать/разрешать Прерывания и т.д. Однако, многие приложения используют фиксированный формат Кадра, фиксированный Бодрейт и настройку Управляющих Регистров и в этом случае инициализация USART может проводиться прямо в Главной Программе или может комбинироваться кодом инициализации других I/O-модулей.

# Передача Данных – Передатчик USART

Работа Передатчика USART разрешается установкой Бита TXEN в Регистре UCSRB. Если это сделано, то нормальная функция порта, где находится ножка TxD, игнорируется, и ей назначается функция последовательной передачи данных от Передатчика USART. Бодрейт, Режим Работы и формат Кадра должны быть настроены перед тем, как начать пользоваться Передачей Данных. Если используется Синхронный Режим Работы, то другие функции ножки XCK будут игнорироваться, а этот Вывод будет использоваться как линия синхронизации при передаче данных.

# Пересылка Кадров размером от 5до 8 Битов Данных

Передача Данных начинается после загрузки в Буфер Передатчика полезных данных, которые надо отправить по линии связи. ЦПУ может заполнить этот Буфер записав соответствующие Полезные Данные в Регистр UDR. Эти Буферизированные Полезные Данные переместятся в Сдвиговый Регистр, когда этот Регистр будет готов отправить новый Кадр. Сдвиговый Регистр загрузит в себя новые Полезные Данные, если линия передачи находится в Холостом Состоянии (Idle) или немедленно после последнего Стопового Бита предыдущего передаваемого кадра. Когда Сдвиговый Регистр загрузится Новыми Полезными Данными, он начнет передавать их с заранее установленной скоростью, скорость (Бодрейт) зависит не только от Регистра UBRR, но и от Бита U2X и XCK, в зависимости от выбранного Режима.

Следующий пример кода содержит простую функцию для Передачи Данных при опросе Флага Опустошения Регистра Данных UDRE. При использовании формата Кадра меньшего, чем 8 бит, то более старшие по номеру разряда биты, которые записываются в Регистр Данных, просто игнорируются. USART должен быть инициализирован перед использованием этих функций. Для Ассемблерной функции предполагается, что передаваемые данные предварительно сохраняются в Регистр R16.

Assembly Code Example[1]

```
USART_Transmit:
  ; Wait for empty transmit buffer
  sbis UCSRA,UDRE
  rjmp USART_Transmit
  ; Put data (r16) into buffer, sends the data
  out  UDR,r16
  ret
```

C Code Example[1]

```
void USART_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
       ;
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

Примечание: 1. Предполагается, что соответствующие заголовочные файлы подключены. _.

Эти функции просто ждут, пока появится Флаг UDRE, что будет символизировать об опустошении Буфера Передатчика. Если Прерывание по Опустошению Регистра UDR разрешено, то при опустошении его произойдет установка Флага UDRE и подпрограмма (функция) запишет Новые Данные в Регистр UDR.

# Пересылка Кадра с 9 Битами Данных

Если используется 9-битный размер Кадра (UCSZ = 7), тогда 9й Бит должен быть записан в Бит TXB8 в Регистре UCSRB до того, как младший **байт** будет записан в Регистр UDR. Следующий пример Кода содержит функцию обработки 9-битного Кадра. Для Ассемблерной функции предполагается, что передаваемые данные предварительно сохраняются в Регистрах R17:R16.

Assembly Code Example(1)(2)

```
USART_Transmit:
  ; Wait for empty transmit buffer
  sbis UCSRA,UDRE
  rjmp USART_Transmit
  ; Copy 9th bit from r17 to TXB8
  cbi  UCSRB,TXB8
  sbrc r17,0
  sbi  UCSRB,TXB8
  ; Put LSB data (r16) into buffer, sends the data
  out  UDR,r16
  ret
```

C Code Example(1)(2)

```
void USART_Transmit( unsigned int data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
       ;
  /* Copy 9th bit to TXB8 */
  UCSRB &= ~(1<<TXB8);
  if ( data & 0x0100 )
    UCSRB |= (1<<TXB8);
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

Примечание: 1. Эти функции могут быть оптимизированы, если содержание Регистра UCSRB сделать статическим. К примеру, можно использовать после инициализации только Бит TXB8 Регистра UCSRB.
2. Предполагается, что соответствующие заголовочные файлы подключены. _.

Девятый Бит может быть использован для индикации адреса Кадра при Мультипроцессорной Связи или для других нужд, к примеру, синхронизации.

# Флаги Передатчика и Прерывания

USART Передатчик имеет два Флага для индикации Его состояния: Флаг UDRE говорит о том, что Регистр UDRE пуст, а Флаг TXC – о том, что Передача Данных завершена. Оба эти флага могут быть использованы для генерирования Прерываний .

Флаг UDRE устанавливается, когда Буфер Передатчика готов снова принять Новые Данные. Этот Бит устанавливается, когда Буфер Передатчика опустошается, а очищается Флаг, когда Буфер принимает Данные для Передачи, которые не были еще перемещены в Сдвиговый Регистр. Для совместимости программного обеспечения с будущими устройствами (МК) всегда устанавливайте этот Бит в ноль при записи в Регистр UCSRA.

При установке Бита UDRIE в Регистре UCSRB происходит разрешение срабатывания Прерывания по опустошению Регистра UDR, это Прерывание будет вызываться так долго, пока Оно будет разрешено (должен быть установлен Глобальный Бит разрешения Прерываний). UDRE очищается при записи данных в UDR. Обработчик Прерывания (функция) должна записать Новые Данные в UDR для того, чтобы снять Флаг UDRE или обработчик может запретить Прерывание по Опустошению, в противном случае прерывание произойдет снова после выхода из Подпрограммы Обработчика.

Флаг TXC устанавливается в 1, когда весь Кадр был «выдвинут» из Сдвигового Регистра, а новые данные в Регистр UDR не поступили. Флаг TXC очищается автоматически, когда по завершению передачи данных вызывается Прерывание или, еще, он может быть очищен записью в него логической единицы. Флаг TXC полезен при Полудуплексной Связи (к примеру, интерфейс RS-485), когда приложение должно по завершению передачи немедленно освободить линию связи и войти в Режим Приемника.

\

Если в Регистре UCSRB установлен Бит TXCIE, то по завершению Передачи Данных будет устанавливаться Флаг TXC и будет вызываться Прерывание (Должно быть установлено Глобальное разрешение Прерываний). Когда срабатывает Прерывание по завершению Передачи Данных, Обработчик Прерывания не должен очищать Флаг TXC, это происходит автоматически при срабатывании Прерывания.

# Генератор Четности (Нечетности) – Генератор Равенства

Генератор Равенства подсчитывает Биты Равенства (Биты Четности и Нечетности) в Кадре данных. Если Бит UPM=1, то Логическая Схема Передатчика вставляет Бит Равенства между последним Битом Данных и Первым Стоповым Битом в Кадре, который следует отослать.

# Выключение Передатчика (запрет)

Выключение Передатчика (установкой Бита TXEN в ноль) не даст эффекта, пока и, если передача данных еще не завершена. _ . Когда же Передатчик будет выключен, он больше не сможет Игнорировать нормальные (обычные) функции вывода TxD.

# Прием Данных – Приемник USART

Работа Приемника USART разрешается установкой Бита RXEN в единицу, который находится в Регистре UCSRB. При разрешении работы Приемника нормальная функция (работа) вывода RxD игнорируется модулем USART и ей назначается функция Входа Приемника с последовательным приемом данных. Настройка Бодрейта, Режима Работы и Формат Кадра должна быть сделана до того, как Приемник будет использоваться по назначению. Если используется Синхронный Режим Работы, то такты на выводе XCK будут использоваться для синхронизации передачи.

# Прием Кадра с 5 и 8 Битами Данных

Приемник начинает принимать Биты Данных, когда успешно определяет приход Стартового Бита. Каждый бит, идущий за стартовым, будет отсемплирован (выбран) в соответствии с предустановленным Бодрейтом, либо в соответствии с синхронизующими тактами на выводе XCK, далее они будут передаваться в Сдвиговый Регистр, пока не будет принят Первый Стоповый Бит. Второй Стоповый Бит Приемником будет игнорироваться. По приходу Первого Стопового Бита весь Кадр оказывается в Сдвиговом Регистре, откуда его содержимое перемещается в Буфер Приемника. Буфер Приемника может быть прочитан через чтение Регистра Данных UDR.

Следующий простой пример кода показывает простую функцию для Приемника USART, которая базируется на опросе флага RXC, который, в свою очередь, говорит об окончании приема. При использовании Размера Кадра меньшего, чем 8 Бит Данных, более старшие биты из UDR будут читаться как нули. Настройка USART должна быть выполнена до того, как подобная функция будет использоваться.

Assembly Code Example[1]

```
USART_Receive:
  ; Wait for data to be received
  sbis UCSRA, RXC
  rjmp USART_Receive
  ; Get and return received data from buffer
  in   r16, UDR
  ret
```

C Code Example[1]

```
unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
       ;
  /* Get and return received data from buffer */
  return UDR;
}
```

Примечание: 1. Предполагается, что соответствующие заголовочные файлы подключены. _.

Эта функция (подпрограмма) просто ожидает, пока данные не появятся в Буфере Приемника, для этого она постоянно проверяет Флаг RXC. После того, как этот Флаг установится в 1, эта функция прочитает Регистр UDR и возвратит его значение.

# Прием Кадра с 9 Битами Данных

Если используется размер Кадра равный 9 Битам (в этом случае в Регистр UCSZ загружается число 7), то в качестве 9 Бита Данных используется Бит RXB8, который находится в Регистре UCSRB. Этот Бит должен быть прочитан раньше остальных более младших битов, которые находятся в UDR. Правильным подходом будет сначала читать Статус-Флаги, такие как: FE, DOR и UPE. Они находятся в Регистре UCSRA. После чего, далее, можно прочитать UDR. Чтение UDR изменит состояние Буфера Приемника FIFO и, следовательно, - Битов TXB8, FE, DOR, и UPE, которые все сохранены в FIFO.

Следующий простой пример кода показывает простую функцию для Приемника USART, которая обрабатывает одновременно 9-й Бит и Биты Статуса.

Assembly Code Example[1]

```
USART_Receive:
  ; Wait for data to be received
  sbis UCSRA, RXC
  rjmp USART_Receive
  ; Get status and 9th bit, then data from buffer
  in   r18, UCSRA
  in   r17, UCSRB
  in   r16, UDR
  ; If error, return -1
  andi r18,(1<<FE)|(1<<DOR)|(1<<UPE)
  breq USART_ReceiveNoError
  ldi  r17, HIGH(-1)
  ldi  r16, LOW(-1)
USART_ReceiveNoError:
  ; Filter the 9th bit, then return
  lsr  r17
  andi r17, 0x01
  ret
```

C Code Example[1]

```
unsigned int USART_Receive( void )
{
  unsigned char status, resh, resl;
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
       ;
  /* Get status and 9th bit, then data */
  /* from buffer */
  status = UCSRA;
  resh = UCSRB;
  resl = UDR;
  /* If error, return -1 */
  if ( status & (1<<FE)|(1<<DOR)|(1<<UPE) )
    return -1;
  /* Filter the 9th bit, then return */
  resh = (resh >> 1) & 0x01;
  return ((resh << 8) | resl);
}
```

Примечание: 1. Предполагается, что соответствующие заголовочные файлы подключены. _.

# Флаг Окончания Приема и Прерывание

Приемник USART имеет один Флаг, который служит в качестве индикатора о состоянии Приемника, это Флаг RXC. . По состоянию Бита RXC можно определить имеются ли непрочитанные Данные в Буфере Приемника. Этот Флаг (Бит) устанавливается в Единицу, когда в Буфере Приемника имеются непрочитанные Данные и устанавливается в Ноль, когда Буфер Приемника пуст (т.е. не содержит Данных). Если работа Приемника запрещена (RXEN = 0) то Буфер Приемника будет опустошен и, следовательно, Бит RXC установится в Ноль.

Если Прерывание по Окончанию Приема разрешено, т.е., если установлен Бит RXCIE в Регистре UCSRB, то Это Прерывание будет срабатывать так долго, как долго Бит RXC будет содержать единицу (если Прерывания не запрещены Глобально). Если для приема данных используется Прерывание по Окончанию Приема, то Подпрограмма (Функция) для обработки Прерывания должна осуществлять чтение Регистра UDR для того, чтобы очищать Флаг RXC, в противном случае, по окончанию обработки Прерывания, при выходе из Подпрограммы обработчика Прерывания, это же Прерывание будет вызвано еще раз.

# Флаг Ошибки по Приему

Приемник USART имеет в своем распоряжении Три Флага Ошибки: Флаг FE – говорит об ошибке Кадра, Флаг DOR – говорит о Переполнении Данных, Флаг UPE – говорит об ошибке Четности (Нечетности). Ко всем этим Флагам имеется доступ через Регистр UCSRA. Все эти Флаги располагаются в Буфере Приемника вместе с Кадром Данных, собственно, о статусе которого, они и сообщают. Чтение Флагов, которые находятся в Буфере Приемника, должно осуществляться раньше чтения Регистра Данных UDR, так. как чтение UDR изменяет состояние Буфера Приемника. Состояние Флагов не может быть изменено программно пользователем. Однако, все Флаги могут быть сброшены в Ноль соответствующей записью в Регистре UCSRA, это сделано для совместимости с будущими реализациями USART. Ни один из этих Флагов не может Генерировать Прерывания.

Флаг FE говорит о том, в каком состоянии находится Первый Стоповый Бит Кадра Данных, который сохранен в Буфере Приемника. Флаг FE содержит Ноль, если Стоповый Бит был успешно прочитан (как Единица) и Флаг FE будет установлен в Единицу, если Стоповый Бит будет некорректным – Нулевым. Этот Флаг может быть использован для определения рассинхронизации при работе Приемника или просто – для протокола, когда нужно убедиться, что Ошибки Кадра не было. На Флаг FE установка Бита USBS (в Регистре UCSRC) не влияет, так как Приемник игнорирует все Биты, кроме Первого Стопового. Для совместимости вашей программы с будущими устройствами, при записи в Регистр UCSRA всегда устанавливайте этот Бит USBS в Ноль.

Флаг DOR индицирует потерю Данных при переполнении Буфера Приемника. Флаг DOR устанавливается при полном заполнении Буфера Приемника (два цифровых слова), когда новое слово находится в Сдвиговом Регистре и его Стартовый Бит определен. Если Флаг DOR установлен в Единицу, то это значит, что было потеряно один или несколько Кадров между последним правильно прочитанным Кадром и читаемым в данный момент времени. Для совместимости программы с будущими устройствами (МК), всегда записывайте в этот Бит Ноль при записи данных в Регистр UCSRA. Флаг DOR очищается аппаратно, когда правильно принятый Кадр перемещается из Сдвигового Регистра в Буфер Приемника.

Флаг UPE (Четности-Нечетности) в состоянии логической единицы говорит о том, что следующий кадр, который находится в Буфере Приемника, имеет ошибку на Четность или Нечетность. Если проверка Четности-Нечетности запрещена, то бит UPE всегда будет читаться как ноль. Для совместимости ПО с будущими МК, всегда сбрасывайте бит UPE в ноль при записи данных в Регистр UCSRA. За деталями обращайтесь к разделу «Вычисление Бита Четности (Нечетности)» на странице115 и – к разделу «Схема Проверки Четности-Нечетности» на странице 123.