Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Южно-Уральский государственный университет
Кафедра «Автоматика и управление»

681.3(07)
В858

Вставская Е.В., Константинов В.И.

МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА СИСТЕМ УПРАВЛЕНИЯ

Конспект лекций

Челябинск
Издательский центр ЮУрГУ
2010

УДК 681.3.01(075.8)
В858

Одобрено
учебно-методической комиссией
приборостроительного факультета

Рецензенты:

Тамбовцев В.И., Гудилин А.Е.

Вставская, Е.В.
В858 **Микропроцессорные средства систем управления**: конспект лекций / Е.В. Вставская, В.И. Константинов – Челябинск: Издательский центр ЮУрГУ, 2010. – 91 с.

В конспекте рассматриваются вопросы программирования микропроцессоров и микроконтроллеров на базе микроконтроллера ATmega8535 фирмы Atmel.

Конспект лекций предназначен для студентов очной формы обучения специальности “Управление и информатика в технических системах”.

УДК 681.3.01(075.8)

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Микропроцессором называется программно-управляемое устройство, осуществляющее процесс обработки цифровой информации и управление им. Микропроцессор реализуется в виде большой (БИС) или сверхбольшой (СБИС) интегральной микросхемы. Микропроцессор выполняет роль процессора в цифровых системах различного назначения.

Главной особенностью микропроцессора является возможность программирования логики работы.

Микроконтроллер (MCU) – микросхема, предназначенная для управления электронными устройствами. Типичный микроконтроллер сочетает в себе функции процессора и периферийных устройств, может содержать ОЗУ и ПЗУ. По сути, это однокристальный компьютер, способный выполнять простые задачи. Использование одной микросхемы, вместо целого набора, как в случае обычных процессоров, применяемых в персональных компьютерах, значительно снижает размеры, энергопотребление и стоимость устройств, построенных на базе микроконтроллеров.

Микропроцессорная система (МПС) представляет собой функционально законченное изделие, состоящее из одного или нескольких устройств, главным образом микропроцессорных: микропроцессора и/или микроконтроллера.

Микропроцессорное устройство (МПУ) представляет собой функционально и конструктивно законченное изделие, состоящее из нескольких микросхем, в состав которых входит микропроцессор; оно предназначено для выполнения определенного набора функций: получение, обработка, передача, преобразование информации и управление.

Основные преимущества микропроцессорных систем по сравнению с цифровыми системами на «жесткой логике».

- Многофункциональность: большее количество функций может быть реализовано на одной элементной базе.
- Гибкость: возможность исправления и модификации программы микропроцессора для реализации различных режимов работы системы.
- Компактность: миниатюрные габариты микросхем и уменьшения их количества по сравнению с реализацией на «жесткой логике» позволяют уменьшить габариты устройств.
- Повышение помехоустойчивости: меньшее количество соединительных проводников способствует повышению надежности устройств.
- Производительность: возможность применения больших рабочих частот и более сложных алгоритмов обработки информации.
- Защита информации: возможность защитить программу микропроцессора от считывания позволяет защитить авторские права разработчиков.

Хотя микропроцессор является универсальным средством для цифровой обработки информации, однако отдельные области применения требуют реализации определенных специфических вариантов их структуры и архитектуры. Поэтому по функциональному признаку выделяются два класса:

микропроцессоры общего назначения и специализированные микропроцессоры. Среди специализированных микропроцессоров наиболее широкое распространение получили микроконтроллеры, предназначенные для выполнения функций управления различными объектами, и цифровые сигнальные процессоры (DSP – Digital Signal Processor), которые ориентированы на реализацию процедур, обеспечивающих необходимое преобразование аналоговых сигналов, представленных в цифровой форме.

Неполный список периферии, которая может присутствовать в микроконтроллерах, включает в себя:

- различные интерфейсы ввода-вывода, такие как UART, I²C, SPI, CAN, USB, ETHERNET;
- аналого-цифровые и цифро-аналоговые преобразователи;
- компараторы;
- широтно-импульсные модуляторы;
- таймеры-счетчики;
- генератор тактовой частоты;
- контроллеры дисплеев и клавиатур;
- массивы встроенной флэш-памяти.

Идея размещения на одном кристалле микропроцессора и периферийных устройств принадлежит инженерам М. Кочрену и Г. Буну, сотрудникам Texas Instruments. Первым микроконтроллером был 4-х разрядный TMS1000 от Texas Instruments, который содержал ОЗУ (32 байта), ПЗУ (1 кбайт), часы и поддержку ввода-вывода. Выпущенный в 1972 году, он имел новую по тем временам возможность – добавление новых инструкций.

В 1976 году (через 5 лет после создания первого микропроцессора) на свет появился первый микроконтроллер фирмы Intel, получивший имя 8048.

Помимо центрального процессора, на кристалле находились 1 кбайт памяти программ, 64 байта памяти данных, два восьмибитных таймера, генератор часов и 27 линий портов ввода-вывода. Микроконтроллеры семейства 8048 использовались в игровых консольных приставках Magnavox Odyssey, в клавиатурах первых IBM PC и в ряде других устройств.

Следующий микроконтроллер Intel 8051, выпущенный в 1980 году, стал поистине классическим образцом устройств данного класса. Этот 8-битный чип положил начало целому семейству микроконтроллеров, которые господствовали на рынке вплоть до недавнего времени.

Аналоги 8051 выпускали советские предприятия в Минске, Киеве, Воронеже, Новосибирске, на них выросло целое поколение отечественных разработчиков.

Большинство фирм производителей микроконтроллеров и сегодня выпускают устройства, основанные на этой архитектуре. Среди них Philips, Microchip, Atmel, Dallas, OKI, Siemens и другие.

Motorola и Zilog

Яркими представителями восьмиразрядных микроконтроллеров явились изделия компаний Motorola (68HC05, 68HC08, 68HC11) и Zilog (Z8).

Motorola длительное время не предоставляла средств, позволяющих дешево и быстро начать работать с ее микроконтроллерами, что явно не способствовало их популярности у некорпоративных разработчиков. Однако стоит заметить, что за рубежом микроконтроллеры от Motorola занимают лидирующее положение на рынке. В нашей стране их популярность не очень высока, возможно, еще в силу отсутствия достаточного количества доступных учебных материалов и средств разработки.

Микроконтроллеры фирмы Zilog, основанной бывшими сотрудниками Intel, еще недавно казавшиеся столь многообещающими, не выдержали гонки в стремительно развивающемся секторе рынка, и сегодня система команд Z8 выглядит достаточно устаревшей.

Microchip. Микроконтроллеры PIC.

Первые значительные перемены произошли с появлением PIC-контроллеров фирмы Microchip. Эти чипы предлагались по рекордно низким ценам, что позволило им в короткий срок захватить значительную часть рынка микроконтроллеров. К тому же, кристаллы от Microchip оказались не уступающими, а нередко и превосходящими микроконтроллеры x51 по производительности и не требовали дорогостоящего программатора.

Вместе с микроконтроллерами появились дешевые комплекты PICSTART, содержащие все, что было нужно для того, чтобы, не имея ни средств, ни навыков работы с PIC-контроллерами, быстро создать и отладить на нем продукт.

Эти микроконтроллеры имели хорошие порты, но все остальное было сделано весьма неудобно. Архитектура оставляла желать лучшего, система команд была крайне ограничена. Тем не менее, PIC-контроллеры остаются популярными в тех случаях, когда требуется создать недорогую систему, не предъявляющую высоких требований по ее управлению.

Scinex

На волне успеха PIC-контроллеров появились очень похожие на них изделия фирмы Scinex. Они обладали уже 52-мя командами против PIC-овских 33-х. Были добавлены хорошие инструкции для работы с памятью, улучшена архитектура, каждая команда выполнялась за один такт, что при прочих равных условиях было вчетверо быстрее, чем у Microchip, и к тому же их тактовая частота достигала 100 МГц.

Столь высокая скорость контроллера позволяет его создателям отказаться от различной периферии – таймеров-счетчиков, регистров сдвига в приемопередатчиках, – все это рекомендуется реализовывать программными средствами, благо быстродействия для этого хватает: внутри – лишь сверхбыстрое ядро, память и порты ввода-вывода.

Atmel

Настоящая революция в мире микроконтроллеров произошла в 1996 году, когда корпорация Atmel представила свое семейство чипов на новом прогрессивном ядре AVR. Более продуманная архитектура AVR, быстродействие, превосходящее контроллеры Microchip, привлекательная ценовая политика способствовали оттоку симпатий многих разработчиков от недавних претендентов на звание контроллера номер 1.

Микроконтроллеры AVR имеют более развитую систему команд, насчитывающую до 133 инструкций, производительность, приближающуюся к 1 Mips (1 МГц), Flash ПЗУ программ с возможностью внутрисхемного перепрограммирования. Многие чипы имеют функцию самопрограммирования. AVR-архитектура оптимизирована под язык высокого уровня Си. Кроме того, все кристаллы семейства совместимы «снизу вверх».

Огромную роль сыграла доступность программного обеспечения и средств поддержки разработки. У Atmel много бесплатно распространяемых программных продуктов. Хорошо известно, что развитые средства поддержки разработок при освоении и знакомстве с любым микроконтроллерным семейством играют не менее значимую роль, чем сами кристаллы. Фирма Atmel уделяет этому вопросу большое внимание. Чрезвычайно удачная и совершенно бесплатная среда разработки AVR Studio, работающая под Windows.

Ведущие сторонние производители выпускают полный спектр компиляторов, программаторов, ассемблеров, отладчиков, разъемов и адаптеров.

Самым популярным способом программирования этих микроконтроллеров являются пять проводков, подсоединенных к параллельному порту персонального компьютера.

Можно считать, что AVR постепенно становится еще одним индустриальным стандартом среди 8-разрядных микроконтроллеров общего назначения. Они легкодоступны в России и отличаются в среднем невысокой стоимостью, успешно конкурируя с изделиями компании Microchip. Все это делает микроконтроллеры Atmel AVR одними из самых привлекательных для обучения.

Toshiba

Отдельного упоминания заслуживают мощные контроллеры фирмы Toshiba. Хотя у них и отсутствует внутренняя память программ, нужен кристалл внешнего ПЗУ, но они имеют хорошо развитую периферию и способны поддерживать модули памяти типа SIMM, используемые в IBM. За рубежом эти контроллеры ставятся в DVD-проигрыватели, CD-проигрыватели, автоответчики, — словом, туда, где надо работать с большими объемами памяти.

ACE

ACE выпускает самые маленькие в мире микроконтроллеры. Это 8-разрядные чипы размерами около 3x4 мм, из 8 выводов 6 – это порты ввода-вывода. По возможностям они похожи на Microchip или AVR, но в очень маленьком корпусе. Им можно сделать минимальное обрамление и поместить в ручку какого-нибудь изделия.

Hitachi

Линия микроконтроллеров H8 фирмы Hitachi – это большая семья микроконтроллеров, включающая H8/300, H8/300H, H8/500 и H8S серии. Основа архитектуры H8 базируется на решениях фирмы DEC и их легендарном компьютере PDP-11. Несколько компаний выпускают для этих микроконтроллеров компиляторы ассемблера и языков высокого уровня. H8 могут быть найдены в цифровых фотокамерах, контроллерах принтеров и различных автоматических подсистемах. Также они трудятся в контроллерных блоках RCX робоконструкторов Lego MindStorm.

Среди крупных производителей микроконтроллеров следовало бы упомянуть также Cypress, Texas Instruments, Dallas Semiconductor, Philips, Infineon (Siemens), STMicroelectronics, Futjitsu, Mitsubishi Electronics, Temic, National Semoconductor, Oki Semiconductor, STelectronics и др.

Классификация микропроцессоров представлена на рис. 1.

Все микроконтроллеры, в свою очередь, можно условно разделить на 3 класса

- 8-разрядные
- 16-разрядные
- 32-разрядные

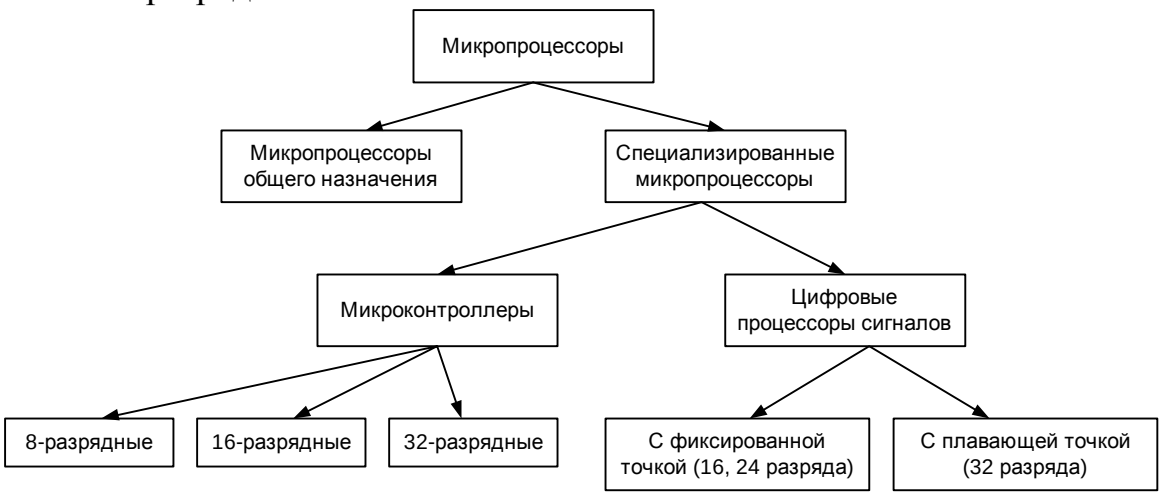


Рис. 1. Классификация современных микропроцессоров по функциональному признаку

8-разрядные микроконтроллеры имеют относительно низкую производительность, которая вполне достаточна для решения широкого круга задач управления различными объектами. Это простые и дешевые микроконтроллеры, ориентированные на использование в относительно несложных устройствах массового выпуска. Основными областями их применения являются бытовая и измерительная техника, промышленная автоматика, автомобильная электроника, теле-, видео- и аудиоаппаратура, средства связи. Для этих микроконтроллеров характерна реализация Гарвардской архитектуры, где используется отдельная память для хранения программ и данных. Для хранения программ в различных типах микроконтроллеров применяется либо масочно-программируемое ПЗУ (ROM), либо однократно-программируемое ПЗУ (PROM), либо электрически перепрограммируемое ПЗУ (EPROM, EEPROM или Flash). Внутренняя память программ обычно имеет объем

от нескольких единиц до десятков килобайт. Для хранения данных используется регистровый блок, организованный в виде нескольких регистровых банков, или внутреннее ОЗУ. Объем внутренней памяти данных составляет от нескольких десятков байт до нескольких килобайт. Ряд микроконтроллеров этой группы позволяет, в случае необходимости, дополнительно подключать внешнюю память команд и данных, объемом до 64…256 килобайт. Микроконтроллеры этой группы обычно выполняют относительно небольшой набор команд (30-100), использующих наиболее простые способы адресации. Такие микроконтроллеры обеспечивают выполнение большинства команд за один такт машинного времени.

16-разрядные микроконтроллеры во многих случаях являются усовершенствованной модификацией своих 8-разрядных прототипов. Они характеризуются не только увеличенной разрядностью обрабатываемых данных, но и расширенной системой команд и способов адресации, увеличенным набором регистров и объемом адресуемой памяти, а также рядом других дополнительных возможностей. Обычно эти микроконтроллеры позволяют расширить объем памяти программ и данных до нескольких мегабайт путем подключения внешних микросхем памяти. Во многих случаях реализуется их программная совместимость с более младшими 8-разрядными моделями. Основная сфера применения таких микроконтроллеров – сложная промышленная автоматика, телекоммуникационная аппаратура, медицинская и измерительная техника.

32-разрядные микроконтроллеры содержат высокопроизводительный процессор, соответствующий по своим возможностям младшим моделям микропроцессоров общего назначения. В ряде случаев процессор, используемый в этих микроконтроллерах, аналогичен CISC- или RISC-процессорам, которые выпускаются или выпускались ранее в качестве микропроцессоров общего назначения. Например, в 32-разрядных микроконтроллерах компании Intel используется процессор i386, в микроконтроллерах компании Motorola широко применяется процессор 68020, в ряде других микроконтроллеров в качестве процессорного ядра служат RISC-процессоры типа PowerPC. На базе данных процессоров были реализованы различные модели персональных компьютеров. Введение этих процессоров в состав микроконтроллеров позволяет использовать в соответствующих системах управления огромный объем прикладного и системного программного обеспечения, созданный ранее для соответствующих персональных компьютеров. Кроме 32-разрядного процессора на кристалле микроконтроллера размещается внутренняя память команд емкостью до десятков килобайт, память данных емкостью до нескольких килобайт, а также сложно-функциональные периферийные устройства – таймерный процессор, коммуникационный процессор, модуль последовательного обмена и ряд других. Микроконтроллеры работают с внешней памятью объемом до 16 Мбайт и выше. Они находят широкое применение в системах управления сложными объектами промышленной автоматики (двигатели, робототехнические устройства, средства комплексной автоматизации производства), в контрольно-измерительной аппаратуре и телекоммуникационном оборудовании. Во внутренней структуре этих микроконтроллеров реализуется Принстонская или Гарвардская архитектура.

Входящие в их состав процессоры могут иметь CISC- или RISC-архитектуру, а некоторые из них содержат несколько исполнительных конвейеров, образующих суперскалярную структуру.

Цифровые сигнальные процессоры (ЦСП, DSP) представляют класс специализированных микропроцессоров, ориентированных на цифровую обработку поступающих аналоговых сигналов. Специфической особенностью алгоритмов обработки аналоговых сигналов является необходимость последовательного выполнения ряда команд умножения-сложения с накоплением промежуточного результата в регистре-аккумуляторе. Поэтому архитектура ЦСП ориентирована на реализацию быстрого выполнения операций такого рода. Набор команд этих процессоров содержит специальные команды MAC (Multiplication with Accumlation), реализующие эти операции. Значения поступившего аналогового сигнала может быть представлено в виде числа с фиксированной или с “плавающей” точкой. В соответствии с этим ЦСП делятся на процессоры, обрабатывающие числа с фиксированной или плавающей точкой. Более простые и дешевые ЦСП с фиксированной точкой обычно обрабатывают 16-разрядные операнды, представленные в виде правильной дроби. Однако ограниченная разрядность в ряде случаев не позволяет обеспечить необходимую точность преобразования. Поэтому в ЦСП с фиксированной точкой, выпускаемых компанией Motorola, принято 24-разрядное представление операндов. Наиболее высокая точность обработки обеспечивается в случае представления данных в формате с «плавающей» точкой. В ЦСП, обрабатывающих данные с «плавающей» точкой, обычно используется 32-разрядный формат их представления. Для повышения производительности при выполнении специфических операций обработки сигналов в большинстве ЦПС реализуется Гарвардская архитектура с использованием нескольких шин для передачи адресов, команд и данных. В ряде ЦПС нашли применение также некоторые черты VLIW-архитектуры: совмещение в одной команде нескольких операций, обеспечивающих обработку имеющихся данных и одновременную загрузку в исполнительный конвейер новых данных для последующей обработки.

При проектировании цифровой системы необходимо осуществить правильный выбор микроконтроллера. Основная цель – выбрать наименее дорогой микроконтроллер (чтобы снизить общую стоимость системы), но в то же время удовлетворяющий спецификации системы, т. е. требованиям по производительности, надежности, условиям применения и т. д.

Основные критерии выбора микроконтроллера представлены ниже в порядке значимости.

- Пригодность для прикладной системы. Может ли она быть сделана на однокристальном микроконтроллере или ее можно реализовать на основе какой-либо специализированной микросхемы.
- Имеет ли микроконтроллер требуемое число контактов, портов ввода-вывода, поскольку в случае их недостатка он не сможет выполнить работу, а в случае избытка цена будет слишком высокой.

- Имеет ли микропонтроллер все требуемые периферийные устройства, такие как аналого-цифровой, цифро-аналоговый преобразователи, интерфейсы связи и т.д.
- Имеет ли микропонтроллер другие периферийные устройства, которые не потребуются в системе (это зачастую увеличивает стоимость микроконтроллера).
- Обеспечивает ли ядро микропонтроллера необходимую производительность, т. е. вычислительную мощность, позволяющую обрабатывать системные запросы в течение всей жизни системы на выбранном прикладном языке.
- Выделено ли в бюджете проекта достаточно средств, чтобы позволить себе использовать данный микроконтроллер. Для ответа на этот вопрос, обычно требуются расценки поставщика. Если данный микроконтроллер не приемлем для проекта, все остальные вопросы становятся несущественными, и разработчик долеж начать поиски другого микроконтроллера.
- Доступность.
    - Существует ли устройство в достаточных количествах.
    - Производится ли оно сейчас.
    - Что ожидается в будущем.
    - Поддержка разработчика.
    - Ассемблеры.
    - Компиляторы.
    - Средства отладки.
    - Внутрисхемные эмуляторы.
    - Отладочные мониторы.
    - Отладчики программ в исходных текстах.
- Информационная поддержка
    - Примеры применения.
    - Сообщения об ошибках.
    - Утилиты, в том числе бесплатные ассемблеры.
    - Примеры исходных текстов.
    - Поддержка применений у поставщика.
    - Квалификация поддерживающего персонала, действительно ли он заинтересован в помощи при решении вашей проблемы.
    - Связь с поддерживающим профессионалом.
- Надежность фирмы производителя.
    - Компетентность, подтвержденная разработками.
    - Надежность производства, т.е. качество продукции.
    - Время работы в этой области.

Чтобы заставить микроконтроллер выполнять то, что мы хотим нужно написать программу для него. Это можно делать на разных языках программирования, но чаще всего используются ассемблер и Си. Все равно в конце получается выходной файл с расширением .hex, который и записывается в микроконтроллер.

Вся информация (электрические параметры, габариты, особенности программирования и т.д.) о микроконтроллерах находится в специальных документах – даташитах (Data Sheet), которые являются своеобразными подробными руководствами для применения микросхем и других электронных приборов. Даташиты обычно можно бесплатно скачивать с сайтов производителей, или со специализированных сайтов.

Еще одна нужная вещь – это так называемые аппноуты (Application Note). Эти документы создают производители микроконтроллеров. В них описывается практическое применение микроконтроллеров, приведены схемы устройств, прошивки, описание работы устройства.

Перед тем, как прошить программу в микросхему, можно промоделировать ее работу на компьютере, для этого существуют различные симуляторы и эмуляторы. В этих программах инженеры рисуют схему устройства, указывают пути к файлам прошивки и смотрят в реальном времени (или не в реальном) на работу устройства. Если что-то не так, корректируют код программы. Такое виртуальное моделирование значительно ускоряет и облегчает процесс написания программ.

В некоторых компиляторах присутствуют отладчики («дебаггеры»), в которых все не так наглядно, но зато найти ошибки в программе гораздо проще. Эти возможности комбинируются в разных средствах разработки.

Отладчики можно разделить на симуляторы и эмуляторы.

Симуляторы — совокупность программных средств, моделирующих работу других программ или их отдельных частей.

Эмуляторы – совокупность программных и аппаратных средств, позволяющих воспроизвести работу других программ или их отдельных частей.

С помощью компьютерно-механических симуляторов, абсолютно точно воспроизводящих интерьер кабины аппарата, тренируются пилоты, космонавты, машинисты высокоскоростных поездов.

ПО для разработки программ для AVR:

AVRStudio – IDE + ассемблер + отладчик

IAR Embedded Workbench for Atmel AVR – компилятор C/C++

CodeVisionAVR – компилятор C + генератор начального кода

Image Craft C (ICC) AVR – компилятор C

PROTEUS – симулятор AVR

Программаторы:

AVReal – программатор подключение через LPT порт, совместим с CodeVisionAVR

PonyProg – программатор подключение через COM порт, поддерживает МК AVR, PIC и др.

AVRISP mkII In-System Programmer – программатор, совместимый с AVR Studio, поддерживает все 8-разрядные микроконтроллеры ATMEL, подключается через порт USB.

ОБЗОР МИКРОКОНТРОЛЛЕРОВ AVR ФИРМЫ ATMEL

Идея разработки нового прогрессивного RISC-ядра зародилась в норвежском городе Тронхейм (Trondheim) в светлых головах двух студентов Norwegian University of Science and Technology (NTNU). Звали изобретателей Альф-Эгил Боген (Alf-Egil Bogen) и Вегард Воллен (Vegard Wollen). Находясь в очаровательном окружении смеси университетских зданий, вычислительных центров и кафе местечка Bakklandet, будущие директора Atmel Norway создали архитектуру, которая стала одной из самых удачных на мировом рынке микроконтроллеров.

В 1995 году Боген и Воллен решили предложить американской корпорации Atmel, известной на тот момент своим «ноу-хау» изготовления чипов с Flash-памятью, выпускать новый 8-битный RISC-микроконтроллер и снабдить его Flash-памятью программ на кристалле. Идея настолько понравилась руководству Atmel Corp., что было принято решение незамедлительно инвестировать данный проект.

Рис. 2. Vegard Wollen

В 1996 году был основан исследовательский центр Atmel в Тронхейме. Стоит сказать, что 150-тысячный Тронхейм усилиями своего университета каждый год порождает до 20-ти новых компаний, специализирующихся в секторах рынка начиная от автоматизации и до передачи и обработки данных. В конце 1996 года был выпущен опытный кристалл AT90S1200, а во второй половине 1997-го корпорация Atmel приступила к серийному производству нового семейства микроконтроллеров, к их рекламной и технической поддержке.

Новое ядро было запатентовано и получило название AVR, которое по прошествии уже нескольких лет стало трактоваться самыми различными способами. Кто-то утверждает, что это не иначе как **A**dvanced **V**irtual **R**ISC, другие полагают, что не обошлось здесь без **A**lf Egil Bogen **V**egard Wollan **R**ISC. Держателями патента при этом являются: Wollan, Vegard (NO); Bogen, Alf-Egil (NO); Myklebust, Gaute (NO); Bryant, John, D. (US).

Система команд и внутреннее устройство чипов AVR разрабатывались совместно с фирмой IAR Systems – производителем компиляторов языков программирования C/C++, что обеспечило уникальные характеристики этих микроконтроллеров. В результате для AVR стало возможным получать высокую плотность кода при использовании языков высокого уровня, практически не теряя в производительности по сравнению с программами, написанными на низкоуровневом языке Ассемблера.

Кроме того, использование прогрессивной технологии конвейеризации у AVR сокращало цикл «выборка – исполнение» команды. Например, у микроконтроллеров семейства x51 короткая команда выполняется за 12 тактов

генератора. В PIC-контроллерах фирмы Microchip, где уже реализован конвейер, короткая команда выполняется за 4 периода тактовой частоты. В микроконтроллерах AVR короткая команда в общем потоке выполнялась всего за один период тактирующего сигнала. Такое построение кристалла обеспечило существенное повышение производительности, которая в пределе может достигать значения 1 Mips (1 миллион операций в секунду) на 1 МГц. Это во многих случаях при заданной производительности позволяло снизить тактовую частоту, а значит, и потребляемую мощность устройства. AVR-микроконтроллеры предоставляли более широкие возможности по оптимизации производительности и энергопотребления, что было особенно важно при разработке приложений с батарейным питанием.

В рамках единой базовой архитектуры микроконтроллеры AVR подразделяются на 3 семейства:

- Tiny AVR – имеют относительно небольшие объемы памяти программ (1…2 кбайт) и весьма ограниченную периферию; практически все они выпускаются в 8-выводных корпусах и предназначены для так называемых «бюджетных» решений, принимаемых в условиях жестких финансовых ограничений. Область применения этих микроконтроллеров — интеллектуальные датчики различного назначения (контрольные, пожарные, охранные), игрушки, зарядные устройства, различная бытовая техника и другие подобные устройства.
- Mega AVR – имеют наиболее развитую периферию, наибольшие среди всех микроконтроллеров AVR объемы памяти программ и данных. Они предназначены для использования в мобильных телефонах, контроллерах различного периферийного оборудования (принтеры, сканеры, современные дисковые накопители, приводы CD-ROM/DVD-ROM и т. п.), сложной офисной технике и т. д.
- Classic AVR – базовая линия микроконтроллеров, которая в связи с переходом в 2001-2002 гг. фирмой Atmel на технологические нормы производства 0,35 мкм замещена микроконтроллерами семейства Mega.

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

Система обозначений микроконтроллеров AVR

Система обозначений микроконтроллера AVR представлена на рис. 3.

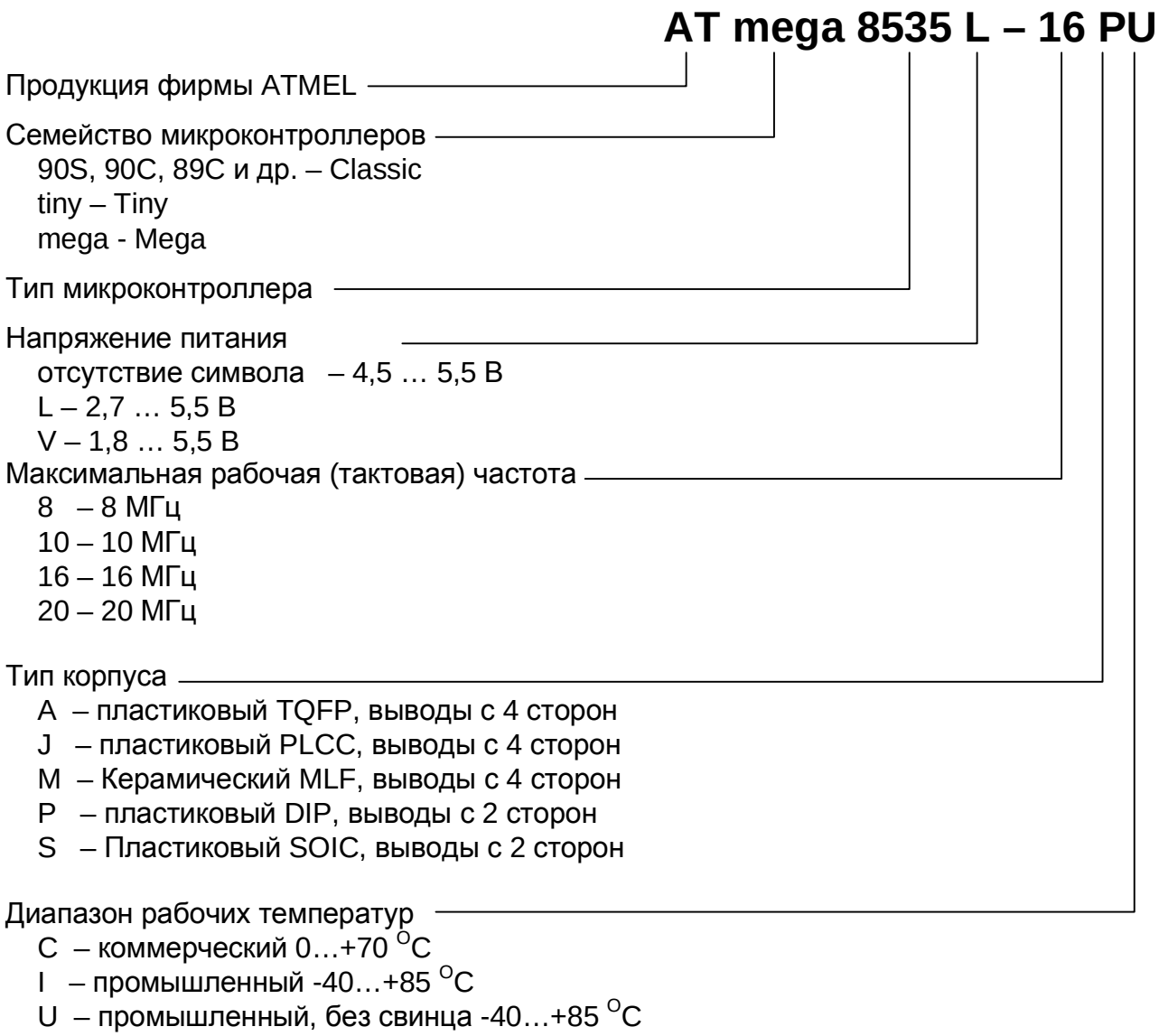


Рис. 3. Система обозначений микроконтроллера AVR

АРХИТЕКТУРА МИКРОКОНТРОЛЛЕРА ATMEGA8535

Архитектура микроконтроллера ATmega8535 включает в себя

- 130 команд процессора (большинство команд – однотактные);
- 32 8-разрядных регистра общего назначения;
- максимальная производительность 16 Mips (миллионов операций в секунду) (максимальная тактовая частота 16 МГц);
- встроенный 2-тактный перемножитель;
- 8 кбайт встроенной электрически перепрограммируемой FLASH памяти (с возможностью самопрограммирования), число циклов стирания-записи памяти не менее 10000;
- 512 байт энергонезависимой памяти EEPROM;
- 512 байт внутреннего ОЗУ (SRAM);
- возможность защиты от чтения и модификации памяти программ и данных;
- возможность программирования непосредственно в системе через последовательный интерфейс SPI.

Периферия:

- 2 8-разрядных независимых таймера-счетчика;
- 1 16-битный таймер-счетчик;
- счетчик реального времени с отдельным тактовым генератором;
- 4 канала ШИМ;
- 8 каналов 10-битного АЦП;
- 2-проводный последовательный интерфейс;
- интерфейс USART;
- последовательный интерфейс SPI;
- встроенный сторожевой таймер;
- встроенный аналоговый компаратор;
- 4 порта ввода-вывода, включающих 32 линии;

Технические характеристики:

- 40(44)- выводный корпус;
- напряжение питания 4,5…5,5 В;
- тактовая частота 0…16 МГц.

Архитектура ядра микроконтроллера ATmega8535

Ядро микроконтроллеров AVR семейства Mega выполнено по усовершенствованной RISC-архитектуре (enhanced RISC) (рис. 4), в которой используется ряд решений, направленных на повышение быстродействия микроконтроллеров.

Арифметико-логическое устройство (АЛУ), выполняющее все вычисления, подключено непосредственно к 32 рабочим регистрам, объединенным в регистровый файл. Благодаря этому, АЛУ может выполнять одну операцию (чтение содержимого регистров, выполнение операции и запись результата обратно в регистровый файл) за такт. Кроме того, практически каждая из команд

(за исключением команд, у которых одним из операндов является 16-битный адрес), занимает одну ячейку памяти программ.

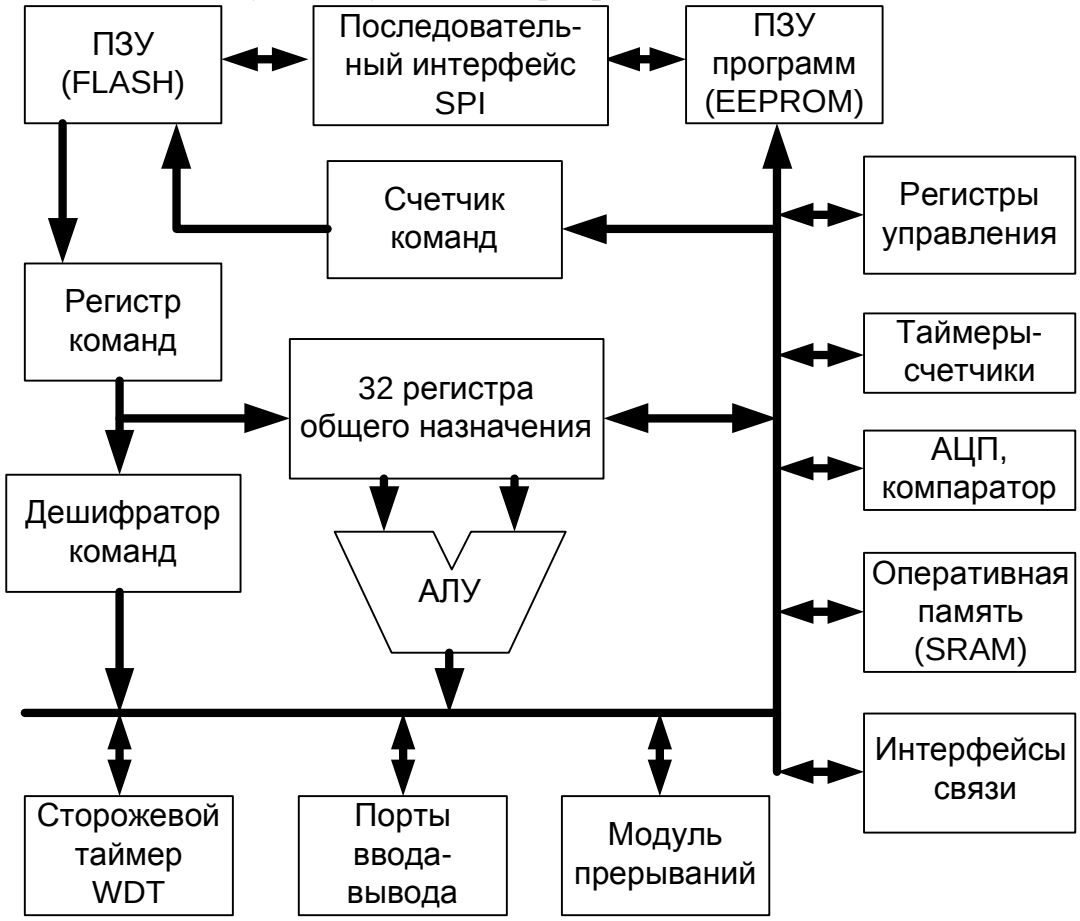


Рис. 4. Упрощенная архитектура ядра микроконтроллера ATmega

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

Арифметико-логическое устройство (АЛУ) поддерживает арифметические и логические операции с операндами в виде двух регистров, либо регистра и константы. Большинство команд процессора выполняются за 1 такт, равный 1 периоду изменения тактового сигнала.

Указатель исполняемой команды хранится в программном счетчике и автоматически изменяет свое значение при переходе на следующую команду. Команды процессора выполняются последовательно, в порядке их следования в программе. При выполнении команд передачи управления (условный или безусловный переход, вызов подпрограммы, прерывание) счетчик команд автоматически изменяет свое значение, указывая на новый адрес исполняемой

команды. Во время прерывания или вызова подпрограммы адрес возврата в вызывающую (основную) программу сохраняется в стеке. Размер стека ограничен размером оперативной памяти.

Цоколевка микроконтроллера ATmega8535

Микроконтроллеры ATmega8535 выпускаются в 44-выводных корпусах типа TQFP, M LF, PLCC, а также в 40-выводных корпусах типа DIP с числом контактов ввода-вывода, равным 32.

Цоколевка микросхемы ATmega8535 приведена на рис. 5, назначения выводов – в табл. 1, где использованы следующие обозначения:

I — вход; O — выход; I/O — вход-выход; P — вывод питания.

Если микросхема имеет неподключенные выводы (NC), подключать их к какому-либо уровню напряжения не рекомендуется.

Таблица 1

Назначения выводов микроконтроллера ATmega8535

Обозначение	Номер вывода			Тип	Описание
	DIP	TQFP MLF	PLCC		
XTAL1	13	8	14	I	Вход тактового генератора
XTAL2	12	7	13	O	Выход тактового генератора
$\overline{\text{RESET}}$	9	4	10	I	Вход сброса
Выводы питания					
AREF	32	29	35	P	Вход опорного напряжения для АЦП
AVCC	30	27	33	P	Вывод источника питания АЦП
VCC	10	5,17, 38	11,2 3, 44	P	Вывод источника питания
GND	11, 31	6,18, 28, 39	12, 24, 34,1	P	Общий вывод
Порт А. 8-битный двунаправленный порт ввода-вывода с внутренними подтягивающими резисторами					
PA0 (ADC0)	40	37	43	I/O	0-й бит порта А. Вход АЦП
PA1 (ADC1)	39	36	42	I/O	1-й бит порта А. Вход АЦП
PA2 (ADC2)	38	35	41	I/O	2-й бит порта А. Вход АЦП
PA3 (ADC3)	37	34	40	I/O	3-й бит порта А. Вход АЦП
PA4 (ADC4)	36	33	39	I/O	4-й бит порта А. Вход АЦП
PA5 (ADC5)	35	32	38	I/O	5-й бит порта А. Вход АЦП
PA6 (ADC6)	34	31	37	I/O	6-й бит порта А. Вход АЦП
PA7 (ADC7)	33	30	36	I/O	7-й бит порта А. Вход АЦП
Порт В. 8-битный двунаправленный порт ввода-вывода с внутренними подтягивающими резисторами					
PB0(T0/XCK)	1	40	2	3	0-й бит порта В. Вход внешнего тактового сигнала таймера/счетчика Т0 Вход/выход внешнего тактового сигнала USART
PB1 (T1)	2	41	3	4	1-й бит порта В. Вход внешнего тактового сигнала таймера/счетчика Т1
PB2 (AIN0/INT2)	3	42	4	5	2-й бит порта В. Неинвертирующий вход компаратора. Вход внешнего прерывания

Окончание табл. 1

Обозначение	Номер вывода			Тип	Описание
	DIP	TQFP MLF	PLCC		
PB3 (AIN1/OCO)	4	43	5	I/O	3-й бит порта B. Инвертирующий вход компаратора Выход таймера-счетчика T0
PB4 ($\overline{SS}$)	5	44	6	I/O	4-й бит порта B. Выбор Slave-устройства на шине SPI
PB5 (MOSI)	6	1	7	I/O	5-й бит порта B. Выход (Master) или вход (Slave) данных модуля SPI
PB6 (MISO)	7	2	8	I/O	6-й бит порта B. Вход (Master) или выход (Slave) данных модуля SPI
PB7 (SCK)	8	3	9	I/O	7-й бит порта B. Выход (Master) или вход (Slave) тактового сигнала модуля SPI
Порт C. 8-битный двунаправленный порт ввода-вывода с внутренними подтягивающими резисторами					
PC0 (SCL)	22	19	25	I/O	0-й бит порта C. Вход/выход тактового сигнала модуля TWI
PC1 (SDA)	23	20	26	I/O	1-й бит порта C. Вход/выход данных модуля TWI
PC2	24	21	27	I/O	2-й бит порта C
PC3	25	22	28	I/O	3-й бит порта C
PC4	26	23	29	I/O	4-й бит порта C
PC5	27	24	30	I/O	5-й бит порта C
PC6 (TOSCl)	28	25	31	I/O	6-й бит порта C Вывод для подключения резонатора к таймеру/счетчику T2
PC7 (TOSC2)	29	26	32	I/O	7-й бит порта C Вывод для подключения резонатора к таймеру-счетчику T2
Порт D. 8-битный двунаправленный порт ввода-вывода с внутренними подтягивающими резисторами					
PDO(RXD)	14	9	15	I/O	0-й бит порта D. Вход USART
PD1 (TXD)	15	10	16	I/O	1-й бит порта D. Выход USART
PD2 (INT0)	16	11	17	I/O	2-й бит порта D. Вход внешнего прерывания
PD3(INT1)	17	12	18	I/O	3-й бит порта D. Вход внешнего прерывания
PD4(OC1B)	18	13	19	I/O	4-й бит порта D. Выход B таймера-счетчика T1
PD5(OC1A)	19	14	20	I/O	5-й бит порта D. Выход A таймера-счетчика T1
PD6(ICP1)	20	15	21	I/O	6-й бит порта D. Вход захвата таймера-счетчика T1
PD7 (OC2)	21	16	22	I/O	7-й бит порта D. Выход таймера-счетчика T2

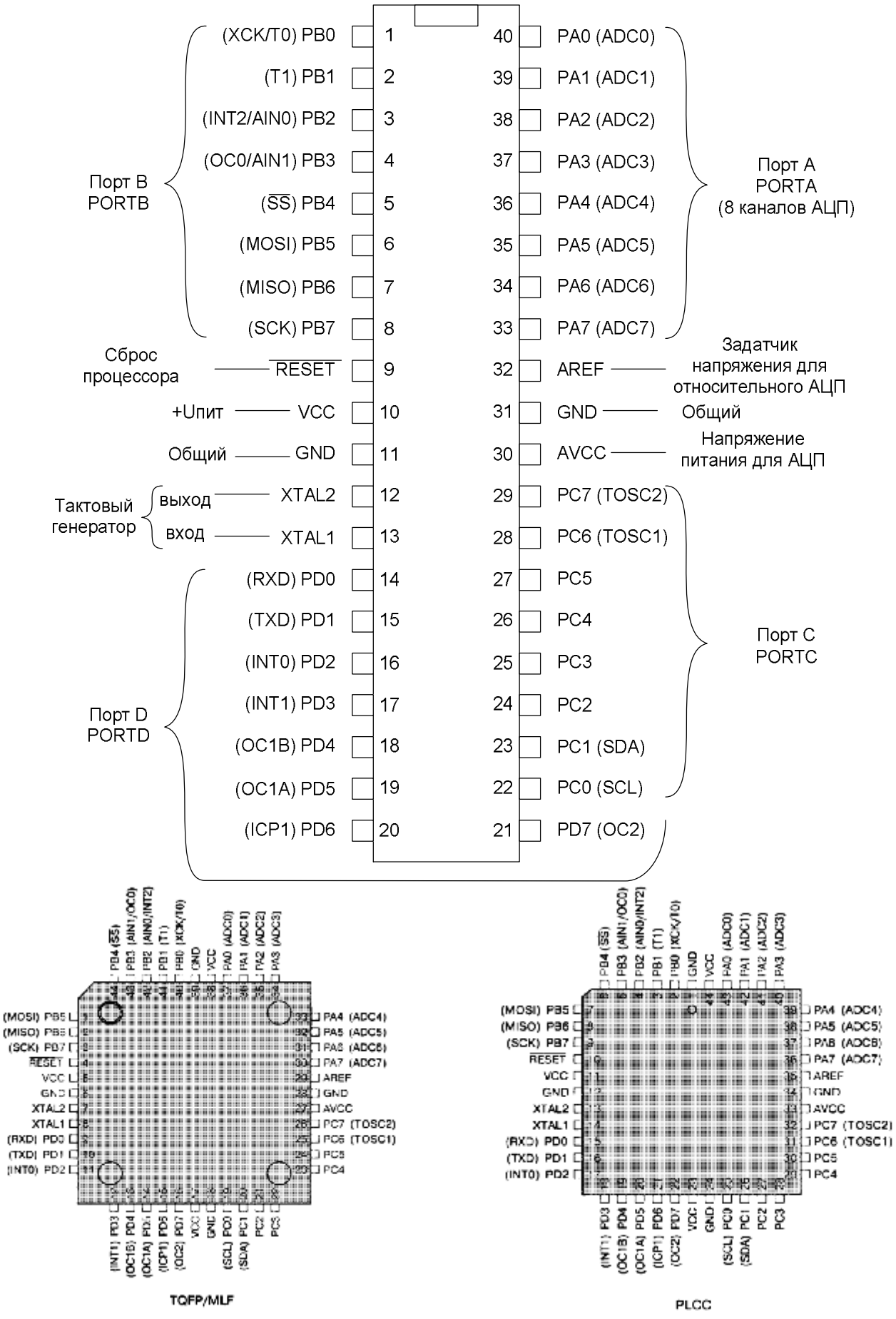


Рис. 5. Цоколевка микроконтроллера ATmega8535

Структурная схема микроконтроллера ATmega8535

Структурная схема микроконтроллера ATmega8535 представлена на рис. 6.

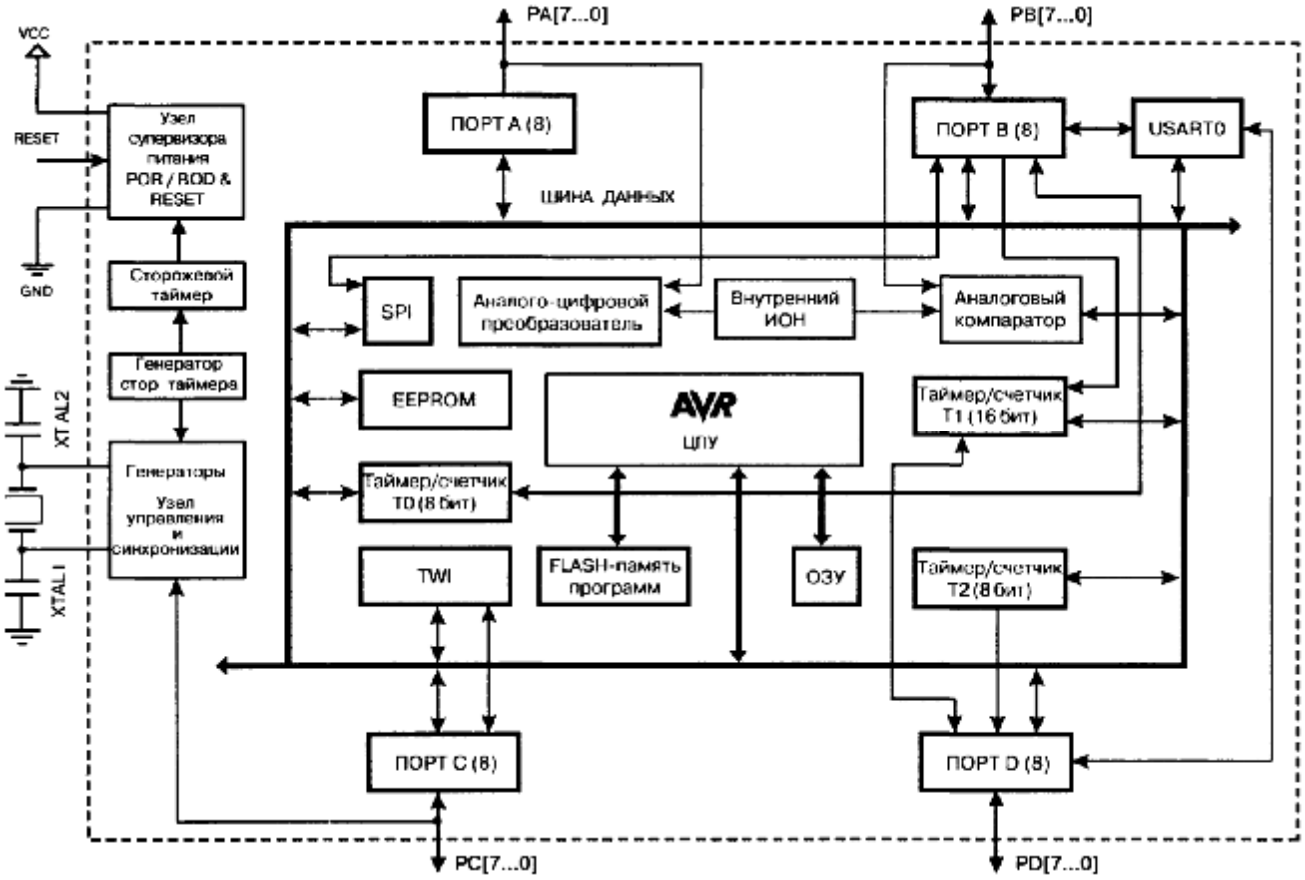


Рис. 6. Структурная схема микроконтроллера ATmega8535

ОРГАНИЗАЦИЯ ПАМЯТИ МИКРОКОНТРОЛЛЕРА ATMEGA8535

Архитектура микропроцессора включает 3 типа памяти:

- память программ FLASH;
- оперативная память (ОЗУ) SRAM;
- энергонезависимая память данных EEPROM.

В микроконтроллерах AVR семейства Mega разделены не только адресные пространства памяти программ и памяти данных, но также и шины доступа к ним. Способы адресации и доступа к этим областям памяти также различны. Такая структура позволяет центральному процессору работать одновременно как с памятью программ, так и с памятью данных, что существенно увеличивает производительность. Каждая из областей памяти данных (ОЗУ и EEPROM) также расположена в своем адресном пространстве.

Память программ

Рис. 7. Память программ

Память программ размером 8 кбайт представляет собой электрически стираемое ППЗУ (FLASH) и организована в виде 4 кбайт × 16 бит с целью поддержки 16-битных команд (рис. 7). Для возможности защиты кода память программ разделена на 2 секции: секция загрузчика (Boot Program) и секция прикладных программ (Application Program). Поскольку микроконтроллеры AVR имеют 16-битную систему команд, объем памяти программ на рис. 7 указан не в байтах, а в 16-битных словах. Память программ является электрически перепрограммируемой, количество циклов перезаписи превышает 10 тысяч.

Микропроцессор поддерживает также внутрисхемное программирование, т. е. прошивку программы в микропроцессор можно осуществлять после монтажа на плату посредством специального разъема программирования.

Для адресации памяти программ используется счетчик команд (Program Counter – PC). Счетчик команд (PC) является 12-битным и адресует все адресное пространство памяти команд.

По адресу 0x0000 памяти программ находится вектор сброса. После инициализации (сброса) микроконтроллера выполнение программы начинается с этого адреса (по этому адресу должна размещаться команда перехода к инициализационной части программы). Начиная с адреса 0x0001 памяти программ, располагается таблица векторов прерываний.

При возникновении прерывания после сохранения в стеке текущего значения счетчика команд происходит выполнение команды, расположенной по адресу

соответствующего вектора. Поэтому по данным адресам располагаются команды перехода к подпрограммам обработки прерываний.

Положение вектора сброса и таблицы векторов прерываний может быть изменено. Они могут располагаться не только в начале памяти программ, как описано выше, но и в начале области загрузчика.

Если в программе прерывания не используются, либо таблица векторов прерываний располагается в области загрузчика, то основная программа может начинаться непосредственно с адреса 0x0001.

Память программ может использоваться не только для хранения кода программы, но и для хранения различных констант. Для пересылки байта из памяти программ в память данных существует специальная команда – LPM.

Оперативная память

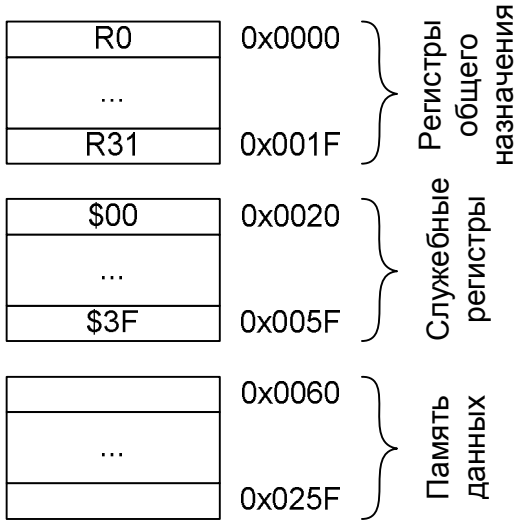


Рис. 8. Оперативная память

Оперативная память состоит из 608 ячеек, разделенных на 3 области (рис. 8):

- 32 регистра общего назначения;
- 64 служебных регистра;
- 512 байт памяти для хранения данных.

Микропроцессор поддерживает 5 способов адресации операндов команд:

- прямая адресация;
- косвенная адресация;
- косвенная адресация со смещением;
- косвенная адресация с предварительным уменьшением адреса;
- косвенная адресация с последующим увеличением адреса.

Рис. 9. Регистры общего назначения

Регистры общего назначения

Все регистры общего назначения (РОН) объединены в регистровый файл быстрого доступа, структура которого показана на рис. 9. В микроконтроллерах AVR все 32 РОН непосредственно доступны АЛУ, в отличие от 8-битных микроконтроллеров других фирм, в которых имеется только один такой регистр – рабочий регистр W (аккумулятор). Благодаря этому любой РОН может использоваться практически во всех командах и как операнд-источник, и как операнд-приемник. Такое решение (в сочетании с конвейерной обработкой) позволяет АЛУ выполнять одну операцию (извлечение операндов из регистрового файла, выполнение команды и запись результата обратно в регистровый файл) за один такт.

Каждый регистр файла имеет свой собственный адрес в пространстве памяти данных. Поэтому к ним можно обращаться двумя способами – как к регистрам и как к памяти, несмотря на то, что физически эти регистры не являются ячейками ОЗУ. Такое решение является еще одной отличительной особенностью архитектуры AVR, повышающей эффективность работы микроконтроллера и его производительность.

Последние 6 регистров файла (R26...R31) могут также объединяться в три 16-битных регистра X, Y и Z (см. рис. 9), используемых в качестве указателей при косвенной адресации памяти данных.

Служебные регистры

Каждый служебный регистр имеет свое имя, адрес и назначение. Краткая характеристика служебных регистров приведена в таблице 2.

Таблица 2

Назначение служебных регистров микроконтроллера ATmega8535

Название	Адрес	Функция
SREG	0x3F(0x5F)	Регистр состояния
SPH	0x3E(0x5E)	Указатель стека, старший байт
SPL	0x3D(0x5D)	Указатель стека, младший байт
OCR0	0x3C(0x5C)	Регистр совпадения таймера/счетчика TO
GICR	0x3B(0x5B)	Общий регистр управления прерываниями
GIFR	0x3A(0x5A)	Общий регистр флагов прерываний
TIMSK	0x39(0x59)	Регистр маски прерываний от таймеров/счетчиков
TIFR	0x38(0x58)	Регистр флагов прерываний от таймеров/счетчиков
SPMCR	0x37 (0x57)	Регистр управления и состояния операций записи в память программ
TWCR	0x36(0x56)	Регистр управления TWI
MCUCR	0x35(0x55)	Регистр управления микроконтроллера
MCUCSR	0x34(0x54)	Регистр управления и состояния микроконтроллера
TCCR0	0x33(0x53)	Регистр управления таймера-счетчика T0
TCNT0	0x32(0x52)	Счетный регистр таймера-счетчика T0
OSCCAL	0x31(0x51)	Регистр калибровки тактового генератора
SFIOR	0x30(0x50)	Регистр специальных функций
TCCR1A	0x2F(0x4F)	Регистр A управления таймера-счетчика T1
TCCR1B	0x2E(0x4E)	Регистр B управления таймера-счетчика T1
TCNT1H	0x2D(0x4D)	Счетный регистр таймера/счетчика T1, старший байт
TCNT1L	0x2C(0x4C)	Счетный регистр таймера/счетчика T1, младший байт
OCR1AH	0x2B (0x4B)	Регистр A совпадения таймера-счетчика T1, старший байт
OCR1AL	0x2A(0x4A)	Регистр A совпадения таймера-счетчика T1, младший байт
OCR1BH	0x29(0x49)	Регистр B совпадения таймера-счетчика T1, старший байт
OCR1BL	0x28(0x48)	Регистр B совпадения таймера-счетчика T1, младший байт
ICR1H	0x27(0x47)	Регистр захвата таймера-счетчика T1, старший байт
ICR1L	0x26(0x46)	Регистр захвата таймера-счетчика T1, младший байт
TCCR2	0x25(0x45)	Регистр управления таймера-счетчика T2
TCNT2	0x24(0x44)	Счетный регистр таймера-счетчика T2
OCR2	0x23(0x43)	Регистр совпадения таймера-счетчика T2
ASSR	0x22(0x42)	Регистр состояния асинхронного режима
VDTCR	0x21(0x41)	Регистр управления сторожевого таймера
UBRRH	0x20 (0x40)	Регистр скорости передачи USART, старший байт
UCSRC		Регистр управления и состояния USART

Окончание табл. 2

Название	Адрес	Функция
EEARH	0x1F(0x3F)	Регистр адреса EEPROM, старший байт
EEARL	0x1E(0x3E)	Регистр адреса EEPROM, младший байт
EEDR	0x1D(0x3D)	Регистр данных EEPROM
EECR	0x1C(0x3C)	Регистр управления EEPROM
PORTA	0x1B(0x3B)	Регистр данных порта A
DDRA	0x1A(0x3A)	Регистр направления данных порта A
PINA	0x19(0x39)	Выводы порта A
PORTB	0x18(0x38)	Регистр данных порта B
DDRB	0x17(0x37)	Регистр направления данных порта B
PINB	0x16(0x36)	Выводы порта B
PORTC	0x15(0x35)	Регистр данных порта C
DDRC	0x14(0x34)	Регистр направления данных порта C
PINC	0x13(0x33)	Выводы порта C
PORTD	0x12(0x32)	Регистр данных порта D
DDRD	0x11(0x31)	Регистр направления данных порта D
PIND	0x00(0x30)	Выводы порта D
SPDR	0x0F(0x2F)	Регистр данных SPI
SPSR	0x0E(0x2E)	Регистр состояния SPI
SPCR	0x0D(0x2D)	Регистр управления SPI
UDR	0x0C(0x2C)	Регистр данных USART
UCSRA	0x0B(0x2B)	Регистр A управления и состояния USART
UCSRB	0x0A(0x2A)	Регистр B управления и состояния USART
UBRRL	0x09(0x29)	Регистр скорости передачи USART, младший байт
ACSR	0x08(0x28)	Регистр управления и состояния аналогового компаратора
ADMUX	0x07(0x27)	Регистр управления мультиплексором АЦП
ADCSRA	0x06(0x26)	Регистр A управления и состояния АЦП
ADCH	0x05(0x25)	Регистр данных АЦП, старший байт
ADCL	0x04(0x24)	Регистр данных АЦП, младший байт
TWDR	0x03(0x23)	Регистр данных TWI
TWAR	0x02(0x22)	Регистр адреса TWI
TWSR	0x01(0x21)	Регистр состояния TWI
TWBR	0x00(0x20)	Регистр скорости передачи TWI

Адресное пространство служебных регистров поддерживает обращение с помощью команд IN (считать данные из регистра) и OUT (записать данные в регистр). Обращение к отдельным битам служебных регистров возможно посредством команд SBI (установить бит), CBI (сбросить бит). Проверка содержимого битов служебных регистров возможна с помощью команд SBIS и SBIC.

К регистрам ввода-вывода можно обращаться и как к ячейкам ОЗУ с помощью соответствующих команд ST/SD/SDD и LD/LDS/LDD (для дополнительных PBB этот способ является единственно возможным).

Среди служебных регистров есть один регистр, используемый наиболее часто в процессе выполнения программ. Это регистр состояния SREG. Он располагается по адресу 0x3F (0x5F) и содержит набор флагов, показывающих текущее состояние микроконтроллера. Большинство флагов автоматически

устанавливаются в «1» или сбрасываются в «0» при наступлении определенных событий (в соответствии с результатом выполнения команд). Все биты этого регистра доступны как для чтения, так и для записи; после сброса микроконтроллера все биты регистра сбрасываются в 0. Эта информация анализируется при выполнении условных переходов. При возникновении прерываний содержимое регистра SREG необходимо сохранять программно. Формат этого регистра показан на рис. 10.

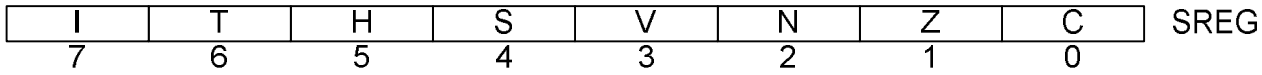


Рис. 10. Регистр состояния SREG

I – бит прерывания; разрешает прерывания, если установлен в «1»; значение «0» запрещает прерывания;

T – источник сохранения/копирования бита; команды копирования бита (BLD – загрузка бита; BST – сохранение бита) используют данный бит в качестве источника;

H – десятичный перенос; устанавливается в «1» при генерации переноса из младшей тетрады битов (4бита) в старшую, используется при работе с двоично-десятичными числами;

S – знаковый бит; определяется как исключающее или битов XOR(V,N);

V – бит переполнения; устанавливается в «1» при выходе за разрядную сетку результата последней команды;

N – бит отрицательного значения; устанавливается в «1» при отрицательном результате выполнения последней команды;

Z – бит нулевого значения; устанавливается в «1» при нулевом результате выполнения последней команды;

C – бит переноса; устанавливается в «1» при генерации переноса последней исполняемой командой.

Память данных

Память данных представляет собой 512 8-битных ячеек, расположенных по адресам 0x0060 …0x025F и поддерживает все 5 видов адресации.

Энергонезависимая память данных

Энергонезависимая память данных EEPROM представляет собой 512 байт, организованных таким образом, что содержимое каждого байта отдельно можно считать или записать. Количество циклов перезаписи энергонезависимой памяти превышает 100 тысяч.

Чтение-запись данных в EEPROM осуществляется посредством использования четырех регистров из области служебных регистров SRAM:

- пары регистров для хранения адреса обращения к ячейке EEPROM EEARH-EEARL содержит 8 бит для обращения к ячейкам адресного пространства EEPROM (512 байт) (рис. 11а);
- регистра приема–передачи данных в EEPROM – EEDR (рис. 11б);
- регистра управления чтением-записью EEPROM – EECR (рис. 11в).

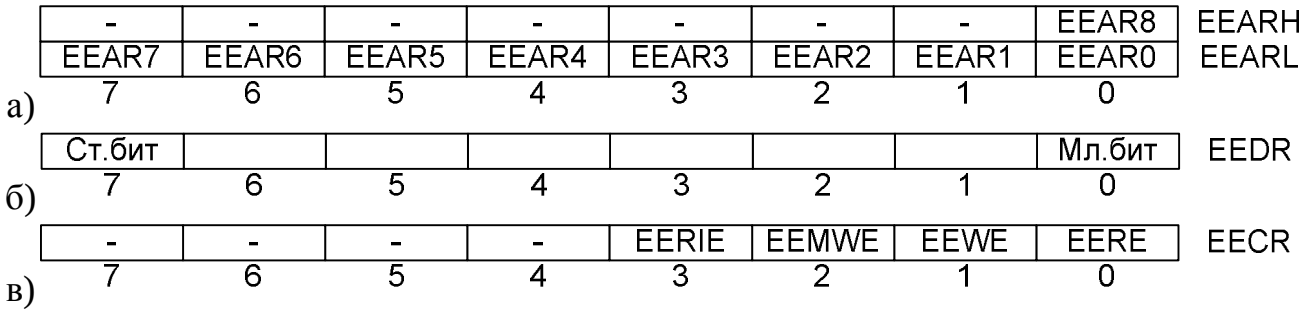


Рис. 11. Регистры, обслуживающие энергонезависимую память микроконтроллера

EERE – бит разрешения чтения. Должен быть установлен в «1», когда загружен адрес обращения к ячейке памяти EEPROM в регистры EEARH-EEARL, по окончании чтения сбрасывается в «0». Считанные данные заносятся в регистр EEDR.

EEWE – бит разрешения записи. Должен быть установлен в «1», когда загружен адрес обращения к ячейке памяти EEPROM в регистры EEARH-EEARL, и данные для записи в регистр EEDR. По окончании чтения сбрасывается в «0».

EEMWE – если установлен в «1», установка бита EEWE осуществляет запись данных в EEPROM, иначе установка бита EEWE не даст результатов.

EERIE – установка бита в «1» разрешает генерировать прерывание после операции чтения, сброс – запрещает генерацию прерывания.

Запись и чтение данных из энергонезависимой памяти осуществляется соответствующими командами.

```
__EEPUT(unsigned char ADR, unsigned char VAL) // запись
  {
   while (EECR & 0x02);
   EEAR = ADR;
   EEDR = VAL;
   EECR = 0x04;
   EECR = 0x02;
   return;
  }

__EEGET(unsigned char VAR, unsigned char ADR) // чтение
  {
   while (EECR & 0x02);    // ожидание окончания записи
   EEAR = ADR;
   EECR = 0x01;
   VAR = EEDR;
   return;
  }
```

РАБОТА С ПОРТАМИ ВВОДА-ВЫВОДА

Порт ввода-вывода – логическое объединение сигнальных линий, через которое принимаются и передаются данные.

Микроконтроллер ATmega8535 имеет 4 порта ввода-вывода: PORTA, PORTB, PORTC, PORTD, каждый из которых состоит из 8 линий. Каждая линия обозначается как Pnx, где n – обозначение порта (A, B, C, D); x – номер бита (линии) в порте (0…7).

Каждый порт ввода-вывода обслуживают 3 служебных регистра:

- PORTn – регистр, содержащий данные (уровни сигналов) на всех линиях порта, используется для записи сигналов в порт (рис. 12а);
- DDRn – регистр направления линий порта: каждая линия порта может быть сконфигурирована как вход (если соответствующий бит регистра DDRn равен 0) или как выход (если соответствующий бит регистра DDRn равен «1») (рис. 12б);
- PINn – регистр, содержащий состояния входов порта; доступен только для чтения, используется при чтении данных из порта (рис. 12в).

а)

PORTn7	PORTn6	PORTn5	PORTn4	PORTn3	PORTn2	PORTn1	PORTn0	PORTn
7	6	5	4	3	2	1	0	

б)

DDRn7	DDRn6	DDRn5	DDRn4	DDRn3	DDRn2	DDRn1	DDRn0	DDRn
7	6	5	4	3	2	1	0	

в)

PINn7	PINn6	PINn5	PINn4	PINn3	PINn2	PINn1	PINn0	PINn
7	6	5	4	3	2	1	0	

Рис. 12. Регистры обслуживания портов ввода-вывода микроконтроллера

Если какая-то линия порта ввода-вывода в схеме не используется, она должна быть определена как вход (соответствующий бит регистра DDRn должен быть равен нулю).

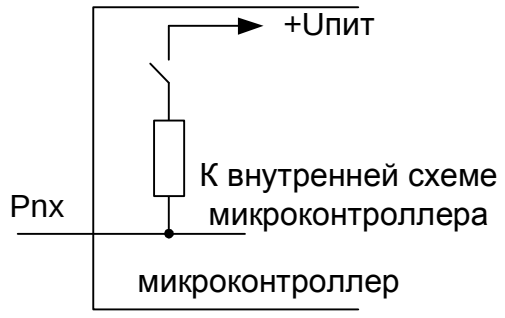


Рис. 13. Подключение подтягивающих резисторов

Большинство линий ввода-вывода могут быть сконфигурированы для выполнения альтернативных функций, обозначенных на цоколевке процессора.

Для всех линий портов ввода-вывода доступна программная конфигурация входных подтягивающих резисторов (рис. 13). **Подтягивающие резисторы** осуществляют доопределение потенциалов «брошенных» входов напряжением высокого уровня (логической «1»).

Отключение подтягивающих резисторов осуществляется установкой в «1» бита PUD регистра SFIOR (рис. 14).

ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10	SFIOR
7	6	5	4	3	2	1	0	

Рис. 14. Регистр SFIOR микроконтроллера

ТАЙМЕРЫ

Таймером называется средство микропроцессора, служащее для измерения времени и реализации задержек. Основой таймера служит суммирующий счетчик, который считает количество импульсов генератора тактовой частоты.

Широтно-импульсная модуляция (ШИМ) (рис. 15) – импульсный сигнал постоянной частоты и переменной **скважности**, то есть отношения периода следования импульса к длительности импульса. С помощью задания скважности (длительности импульсов) можно менять среднее напряжение на выходе ШИМ.

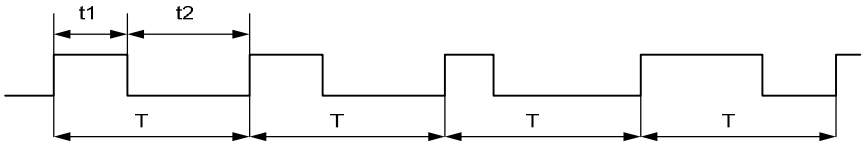


Рис. 15. Широтно-импульсная модуляция

Разрядностью ШИМ называется разрядность двоичного счетчика, используемого для формирования ШИМ-сигнала.

В микроконтроллерах ATmega доступны два основных режима работы ШИМ: быстрый ШИМ и фазовый ШИМ.

Быстрый ШИМ (рис. 16). Период ШИМ определяется максимальным значением, до которого считает счетчик. В этот момент ШИМ-сигнал устанавливается в «1». При достижении счетчиком значения, поданного на второй вход цифрового компаратора, осуществляется сброс выходного ШИМ-сигнала.

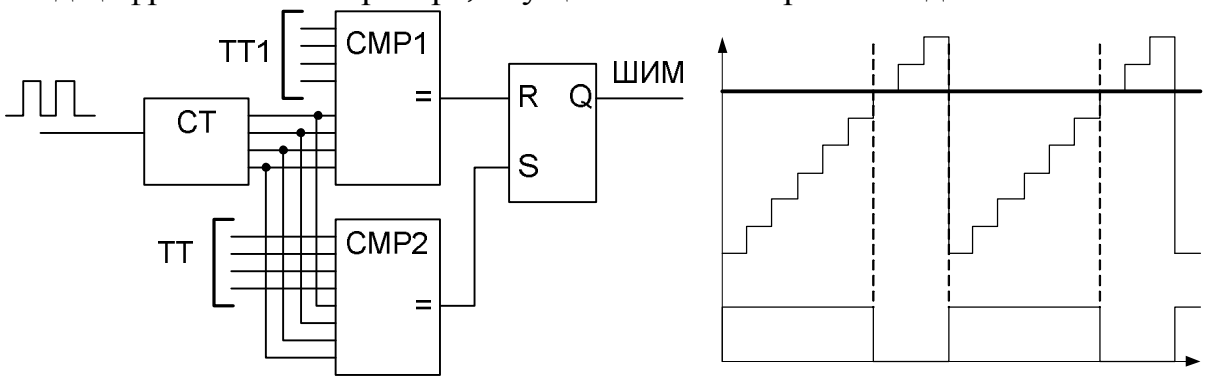


Рис. 16. Быстрый ШИМ

Фазовый ШИМ (рис. 17). В данном режиме счетчик работает как суммирующий и считает от 0 до максимального значения, а при достижении максимального значения работает как вычитающий, считая до 0.

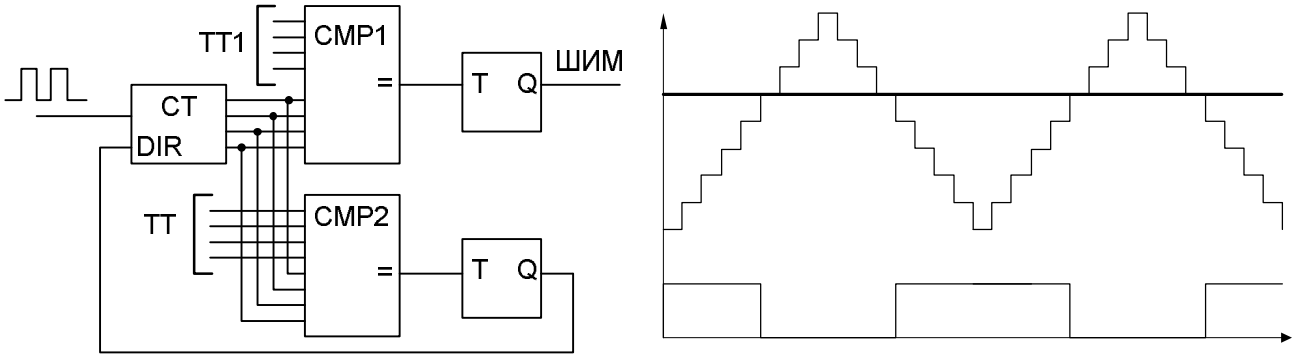


Рис. 17. Фазовый ШИМ

При совпадении значения счетчика с некоторым установленным значением, происходит переключение выхода ШИМ.

Предварительный делитель – делитель частоты тактового сигнала, работающий как один или несколько последовательно соединенных Т-триггеров. Таймер изменяет свое значение на 1 каждые n сигналов тактового импульса. n называют **коэффициентом предварительного деления**.

Микроконтроллер ATmega8535 содержит три таймера общего назначения: два 8-разрядных таймера T0 и T2 и один 16-разрядный таймер T1.

Прерывания таймера

Таймер может генерировать прерывание при переполнении, то есть переходе из максимального значения в нулевое, и при совпадении значений регистров TCNTn и OCRn.

Разрешение и запрет прерываний от таймеров микроконтроллера осуществляется соответствующими битами регистра TIMSK (рис. 18). Единичное значение бита разрешает соответствующее прерывание, нулевое значение бита – запрещает (маскирует) его. Такое управление реакцией микропроцессора на прерывания называется **маскированием прерываний**, а значение соответствующего регистра – **маской прерываний**. Для того чтобы микропроцессор реагировал на возникновение немаскированных прерываний необходимо, чтобы бит I регистра SREG (см. рис. 10) был установлен в «1».

OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
7	6	5	4	3	2	1	0	

Рис. 18. Регистр TIMSK микроконтроллера

OCIE2 – прерывание по совпадению таймера T2;
TOIE2 – прерывание по переполнению таймера T2.
TICIE1 – прерывание по захвату таймера T1.
OCIE1A – прерывание по совпадению таймера T1;
OCIE1B – прерывание по совпадению таймера T1;
TOIE1 – прерывание по переполнению таймера T1.
OCIE0 – прерывание по совпадению таймера T0;
TOIE0 – прерывание по переполнению таймера T0.
Регистр TIFR отвечает за возникновение прерываний.

При наступлении события соответствующий флаг в регистре TIFR (рис. 19) устанавливается в единичное состояние. Расположение битов флагов в регистре TIFR аналогично расположению соответствующих битов разрешения прерывания в регистре TIMSK.

OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
7	6	5	4	3	2	1	0	

Рис. 19. Регистр TIFR микроконтроллера

При запуске программы обработки прерывания он аппаратно сбрасывается в 0.

8-битный таймер-счетчик T0

8-битный таймер-счетчик T0 позволяет осуществлять:

- измерение промежутков времени;
- подсчет внешних событий;
- преобразование выходного сигнала контроллера в сигнал с широтно-импульсной модуляцией (ШИМ);

Структурная схема таймера-счетчика T0 представлена на рис. 20.

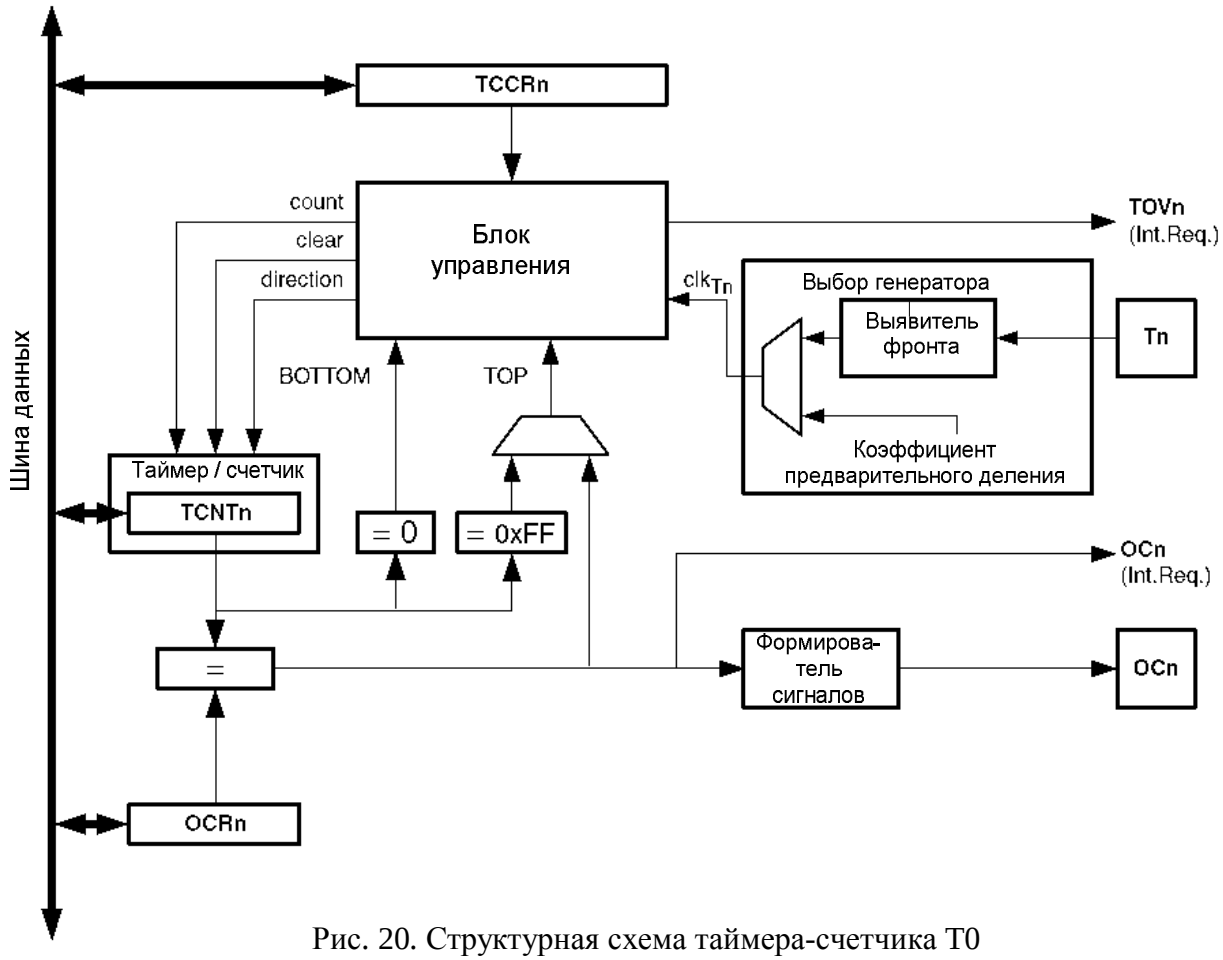


Рис. 20. Структурная схема таймера-счетчика T0

После подачи напряжения питания в регистре TCNTn находится нулевое значение. Регистр TCNTn изменяет свое значение на 1 с приходом каждого тактового сигнала. Тактовый сигнал может быть внутренним (сигнал с генератора, поступающий через предварительный делитель) или внешним, поступающим на соответствующий вход микроконтроллера PB0/T0 (вывод 1).

Счетный регистр таймера-счетчика TCNT0 входит в состав основного блока модуля – блока реверсивного счетчика. В зависимости от режима работы модуля содержимое счетного регистра сбрасывается, инкрементируется или декрементируется по каждому импульсу тактового сигнала таймера-счетчика. Независимо от того, присутствует тактовый сигнал или нет, регистр доступен в любой момент времени как для чтения, так и для записи. Любая операция записи в счетный регистр блокирует работу блока сравнения на время одного периода тактового сигнала таймера-счетчика.

Регистр сравнения OCR0 входит в состав блока сравнения модуля. Во время работы таймера-счетчика производится непрерывное (в каждом такте) сравнение этого регистра с регистром TCNT0. В случае равенства содержимого этих регистров в следующем такте устанавливается флаг OCF0 в соответствующем регистре флагов и генерируется прерывание (если оно разрешено). При совпадении значений регистров может меняться состояние вывода OC0/PB3 микроконтроллера.

Управление таймером-счетчиком T0 осуществляет регистр TCCR0 (рис. 21):

FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
7	6	5	4	3	2	1	0	

Рис. 21. Регистр TCCR0 микроконтроллера

FOC0 – принудительное изменение состояния вывода OC0/PB3 (вывод 4), в режиме ШИМ игнорируется и должен быть равен 0.

WGM01:WGM00 – режим работы таймера:

0 0 – нормальный режим;

0 1 – фазовый ШИМ;

1 0 – сброс при совпадении значений регистров TCNT0 и OCR0;

1 1 – быстрый ШИМ.

COM01:COM00 – подключение вывода OC0/PB3 (при единичном FOC0):

В режиме сброса при совпадении:

0 0 – вывод OC0/PB3 отключен и работает как линия порта ввода-вывода;

0 1 – переключает OC0/PB3 в противоположное состояние при совпадении значений регистров TCNT0 и OCR0;

1 0 – сбрасывает OC0/PB3 в 0 при совпадении значений регистров TCNT0 и OCR0;

1 1 – устанавливает OC0/PB3 в «1» при совпадении значений регистров TCNT0 и OCR0.

В режиме фазового ШИМа:

0 0 – вывод OC0/PB3 отключен и работает как линия порта ввода-вывода;

0 1 – зарезервирован;

1 0 – сбрасывает OC0/PB3 в «0» при совпадении значений регистров TCNT0 и OCR0 при счете на увеличение, устанавливает OC0/PB3 в «1» при счете на уменьшение;

1 1 – устанавливает OC0/PB3 в «1» при совпадении значений регистров TCNT0 и OCR0 при счете на увеличение, сбрасывает OC0/PB3 в «0» при счете на уменьшение.

В режиме быстрого ШИМа:

0 0 – вывод OC0/PB3 отключен и работает как линия порта ввода-вывода;

0 1 – зарезервирован;

1 0 – сбрасывает OC0/PB3 при совпадении значений регистров TCNT0 и OCR0, устанавливает OC0/PB3 при TCNT0=TOP;

1 1 – устанавливает OC0/PB3 при совпадении значений регистров TCNT0 и OCR0, сбрасывает OC0/PB3 при TCNT0=TOP.

CS02:CS00 – управление предварительным делителем

0 0 0 – таймер отключен;

0 0 1 – коэффициент предварительного деления равен 1;

0 1 0 – коэффициент предварительного деления равен 8;

0 1 1 – коэффициент предварительного деления равен 64;

1 0 0 – коэффициент предварительного деления равен 256;

1 0 1 – коэффициент предварительного деления равен 1024;

1 1 0 – подключен внешний тактовый сигнал T0, активен передний фронт;

1 1 1 – подключен внешний тактовый сигнал T0, активен задний фронт.

Режимы работы таймера T0. Простейшим режимом работы таймера является его нормальный режим, при котором таймер работает как суммирующий счетчик, считающий входные импульсы. При переполнении 8-битной разрядной сетки значение счетного регистра TCNT0 сбрасывается. При этом устанавливается бит прерывания таймера TOV0.

Режим сброса при совпадении значений регистров TCNT0 и OCR0. Значение счетного регистра TCNT0 сбрасывается при равенстве его содержимого содержимому OCR0. Таким образом, регистр OCR0 задает максимальную величину, до которой считает таймер. Бит прерывания OCF0 устанавливается каждый раз при совпадении значений TCNT0 и OCR0. Если значение регистра OCR0 меньше значения TCNT0, то счетный регистр сбрасывается при переполнении, после чего считает до значения, равного OCR0. Данный режим может использоваться для генерации меандровых импульсов с частотой, равной

$$f = \frac{f_{OSC}}{2 \cdot N \cdot (1 + OCR0)},$$

где N – коэффициент предварительного деления для таймера.

Быстрый ШИМ представляет собой генератор высокочастотного ШИМ сигнала с частотой

$$f = \frac{f_{OSC}}{256 \cdot N}$$

В общем случае частота быстрого ШИМ может быть в 2 раза выше, чем частота фазового ШИМ. Счетный регистр TCNT0 считает от 0 до максимального значения, после чего сбрасывается. Быстрый ШИМ может работать в неинвертирующем и инвертирующем режимах. В неинвертирующем режиме при совпадении значений TCNT0 и OCR0 состояние вывода OC0 сбрасывается в 0, а при переполнении счетного регистра устанавливается в «1». В инвертирующем режиме при совпадении значений TCNT0 и OCR0 состояние вывода OC0 устанавливается в «1», а при переполнении счетного регистра сбрасывается в «0». Бит прерывания TOV0 устанавливается каждый раз при переполнении счетчика.

Фазовый ШИМ имеет бо̲льшую разрешающую способность по сравнению с быстрым ШИМ. Счетчик считает попеременно в возрастающем и убывающем порядке: от 0 до максимума, затем – от максимума до нуля. В неинвертирующем режиме вывод OC0 устанавливается в «1» при совпадении TCNT0 и OCR0 при суммирующем счете и сбрасывается в «0» при совпадении TCNT0 и OCR0 при вычитающем счете. В инвертирующем режиме вывод OC0 работает инверсно. Флаг прерывания TOV0 устанавливается каждый раз при достижении счетным регистром максимального значения. Частота ШИМ в фазовом режиме определяется как $f = \frac{f_{OSC}}{510 \cdot N}$.

16-битный таймер-счетчик T1

В состав 16-битного таймера-счетчика входят следующие регистры ввода-вывода:

- 16-битный счетный регистр TCNT1;

- 16-битный регистр захвата ICR1;
- два 16-битных регистра сравнения OCR1A, OCR1B;
- два 8-битных регистра управления TCCR1A, TCCR1B.

Упрощенная структурная схема таймера-счетчика T1 представлена на рис. 22.

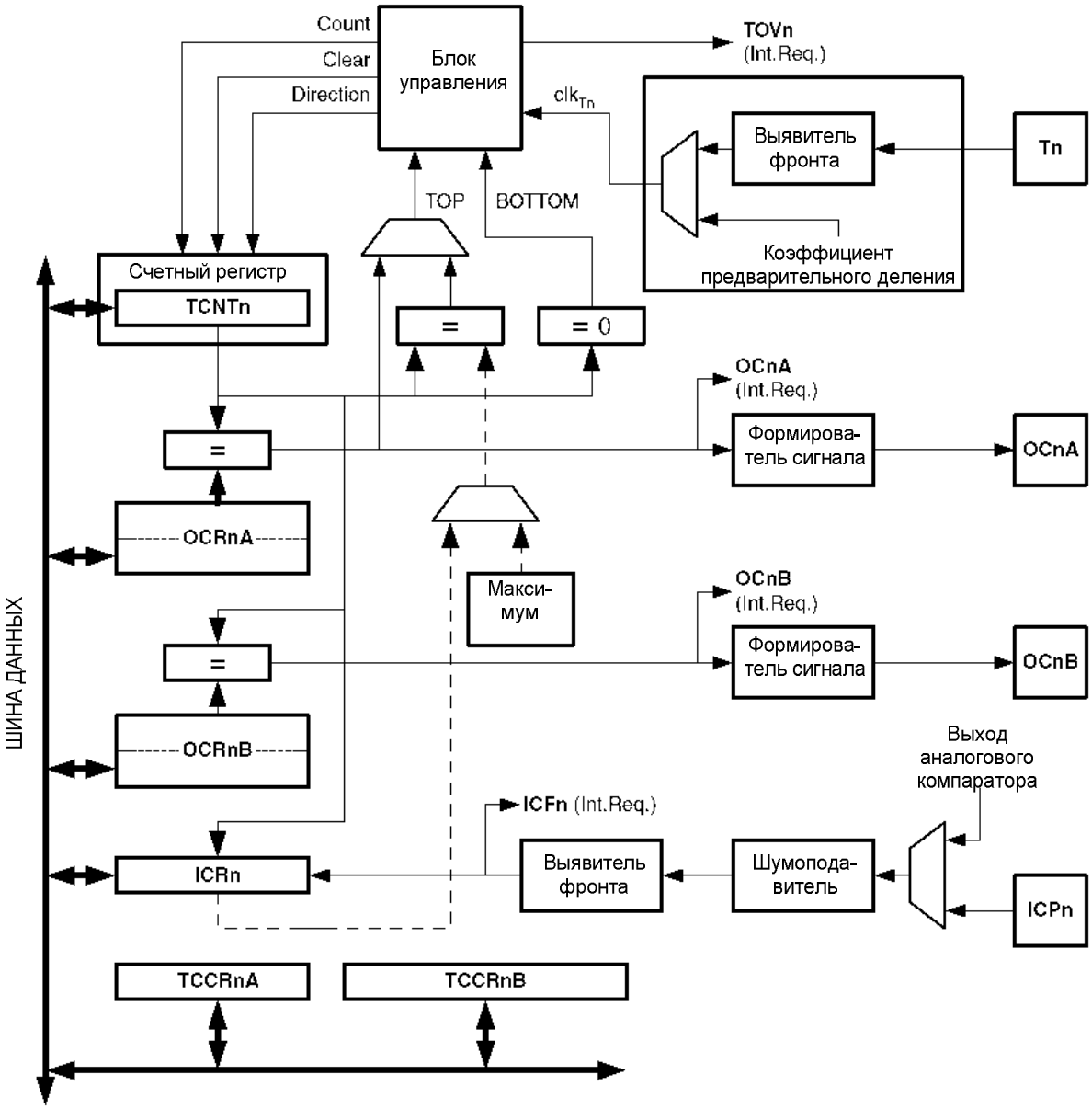


Рис. 22. Упрощенная структурная схема таймера T1 микроконтроллера

Каждый 16-битный регистр таймера-счетчика размещается в двух регистрах ввода-вывода, названия которых получаются добавлением к названию регистра буквы «H» (старший байт) и «L» (младший байт). Счетный регистр таймера-счетчика Tl TCNT1 размещается в регистрах TCNT1H:TCNT1L. Этот регистр входит в состав основного блока модуля – блока реверсивного счетчика. В зависимости от режима работы модуля содержимое счетного регистра сбрасывается, инкрементируется или декрементируется по каждому импульсу тактового сигнала таймера-счетчика. Независимо от того, присутствует тактовый сигнал или нет, регистр доступен в любой момент времени как для чтения, так и для записи. При этом любая операция записи в счетный регистр блокирует работу

всех блоков сравнения на время одного периода тактового сигнала таймера-счетчика.

После подачи напряжения питания в регистре TCNT1 находится нулевое значение. Во время работы таймера-счетчика производится непрерывное (в каждом такте) сравнение этих регистров с регистром TCNT1. В случае равенства содержимого регистра сравнения и счетного регистра, в следующем такте устанавливается флаг OCF1A/OCF1B, в регистре флагов TIFR и генерируется прерывание (если оно разрешено). Также при наступлении этого события может изменяться состояние вывода OC1A/OC1B микроконтроллера. Чтобы таймер-счетчик мог управлять состоянием какого-либо из этих выводов, соответствующий вывод должен быть сконфигурирован как выходной (соответствующий бит регистра DDRn должен быть установлен в «1»). Особенностью работы блока сравнения в режимах, предназначенных для формирования ШИМ-сигналов, является двойная буферизация записи в регистры сравнения. Она заключается в том, что записываемое число на самом деле сохраняется в специальном буферном регистре. А изменение содержимого регистра сравнения происходит только при достижении счетчиком максимального значения.

Регистр захвата ICR1 входит в состав блока захвата, назначение которого – сохранение в определенный момент времени состояния таймера-счетчика в регистре захвата. Это действие может производиться либо по активному фронту сигнала на выводе ICP микроконтроллера, либо по сигналу от аналогового компаратора. Одновременно с записью в регистр захвата устанавливается флаг ICF1 регистра флагов TIFR и генерируется запрос на прерывание. Разрешение прерывания осуществляется установкой в «1» бита TICIE1 регистра маски.

Программно запись в регистр ICR1 возможна только в режимах, в которых регистр захвата определяет модуль счета таймера-счетчика. Вывод ICP в этих режимах отключен от микроконтроллера, а функция захвата соответственно выключена.

Управление таймером-счетчиком T1 осуществляют регистры TCCR1A, TCCR1B (рис. 23):

COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
7	6	5	4	3	2	1	0	

Рис. 23. Регистры TCCR1A и TCCR1B микроконтроллера

COM1A1:COM1A0, COM1B1:COM1B0 – подключение выводов OC1A/PD5, OC1B/PD4 (при единичном FOC1A, FOC1B):

В режиме сброса при совпадении:

0 0 – OC1A/OC1B отключен и работает как линия порта ввода-вывода;

0 1 – переключает OC1A/OC1B в противоположное состояние при совпадении значений TCNT1A/TCNT1B и OCR1A/OCR1B;

1 0 – сбрасывает OC1A/OC1B в «0» при совпадении значений регистров TCNT1A/TCNT1B и OCR1A/OCR1B;

1 1 – устанавливает OC1A/OC1B в «1» при совпадении значений регистров TCNT1A/TCNT1B и OCR1A/OCR1B.

В режиме быстрого ШИМа:

0 0 – OC1A/OC1B отключен и работает как линия порта ввода-вывода;

0 1 – WGM13 = 0 – нормальная работа, OC1A/OC1B отключен;

WGM13=1 – переключает OC1A при совпадении, OC1B – зарезервирован;

1 0 – сбрасывает OC1A/OC1B при совпадении значений регистров TCNT1A/TCNT1B и OCR1A/OCR1B, устанавливает OC1A/OC1B при TCNT1A/TCNT1B = TOP;

1 1 – устанавливает OC1A/OC1B при совпадении значений регистров TCNT1A/TCNT1B и OCR1A/OCR1B, сбрасывает OC1A/OC1B при TCNT1A/TCNT1B = TOP.

В режиме фазового ШИМа:

0 0 – OC1A/OC1B отключен и работает как линия порта ввода-вывода;

0 1 – WGM13 = 0 – нормальная работа, OC1A/OC1B отключен; WGM13 = 1 переключает OC1A при совпадении, OC1B – зарезервирован;

1 0 – сбрасывает OC1A/OC1B при совпадении значений регистров TCNT1A/TCNT1B и OCR1A/OCR1B в режиме суммирования, устанавливает OC1A/OC1B при совпадении в режиме вычитания;

1 1 – устанавливает OC1A/OC1B при совпадении значений регистров TCNT1A/TCNT1B и OCR1A/OCR1B в режиме суммирования, сбрасывает OC1A/OC1B при совпадении в режиме вычитания.

FOC1A, FOC1B – принудительное изменение состояния вывода OC1A/OC1B, в режиме ШИМ игнорируется и должен быть равен «0».

WGM13:WGM10 – режим работы таймера:

0 0 0 0 – нормальный режим;

0 0 0 1 – 8-битный фазовый ШИМ;

0 0 1 0 – 9-битный фазовый ШИМ;

0 0 1 1 – 10-битный фазовый ШИМ;

0 1 0 0 – сброс при совпадении значений регистров TCNT1A и OCR1A;

0 1 0 1 – 8-битный быстрый ШИМ;

0 1 1 0 – 9-битный быстрый ШИМ;

0 1 1 1 – 10-битный быстрый ШИМ;

1 0 0 0 – фазо-частотный ШИМ с модулем пересчета ICR1;

1 0 0 1 – фазо-частотный ШИМ с модулем пересчета OCR1A;

1 0 1 0 – фазовый ШИМ с модулем пересчета ICR1;

1 0 1 1 – фазовый ШИМ с модулем пересчета OCR1A;

1 1 0 0 – сброс при совпадении значений регистров TCNT1A и ICR1;

1 1 0 1 – зарезервировано;

1 0 1 0 – быстрый ШИМ с модулем пересчета ICR1;

1 0 1 1 – быстрый ШИМ с модулем пересчета OCR1A;

ICNC1 – управление схемой подавления помех блока захвата. Если бит сброшен в «0», схема подавления помех выключена (захват производится по первому активному фронту). Если бит установлен в «1», схема подавления помех включена и захват осуществляется только в случае четырех одинаковых выборок, соответствующих активному фронту сигнала

ICES1 – выбор активного фронта сигнала захвата. Если бит сброшен в «0», сохранение счетного регистра в регистре захвата осуществляется по спадающему фронту сигнала. Если бит установлен в «1», то сохранение счетного регистра в регистре захвата осуществляется по нарастающему фронту сигнала. Одновременно с сохранением счетного регистра устанавливается также флаг прерывания ICF1 регистра флагов.

CS12:CS10 – управление предварительным делителем

0 0 0 – таймер отключен;

0 0 1 – коэффициент предварительного деления равен 1;

0 1 0 – коэффициент предварительного деления равен 8;

0 1 1 – коэффициент предварительного деления равен 64;

1 0 0 – коэффициент предварительного деления равен 256;
1 0 1 – коэффициент предварительного деления равен 1024;
1 1 0 – подключен внешний тактовый сигнал T1, активен передний фронт;
1 1 1 – подключен внешний тактовый сигнал T1, активен задний фронт.

Режимы работы таймера T1

Простейшим режимом работы таймера является его нормальный режим, при котором таймер работает как суммирующий счетчик, считающий входные импульсы. При переполнении 16-битной разрядной сетки значение счетного регистра TCNT1 сбрасывается. При этом устанавливается бит прерывания таймера TOV1.

Режим сброса при совпадении. Регистр OCR1A или ICR1 определяют разрешающую способность счетного регистра. Счетчик TCNT1 сбрасывается при достижении одного из значений регистров OCR1A или ICR1 (в зависимости от режима). Бит прерывания OCF1 устанавливается каждый раз при совпадении TCNT1 с соответствующим регистром OCR1A или ICR1. Данный режим может использоваться для генерации меандровых импульсов с частотой, равной

$$f = \frac{f_{OSC}}{2 \cdot N \cdot (1 + OCR1A[ICR1])},$$

где N – коэффициент предварительного деления для таймера.

Быстрый ШИМ представляет собой генератор высокочастотного ШИМ сигнала. Разрядность ШИМ может быть фиксирована (8, 9, 10 бит) или определяться одним из регистров OCR1A или ICR1 и может определяться как

$$R_{PWM} = \frac{\lg(TOP + 1)}{\lg(2)},$$

где TOP – модуль пересчета. Частота ШИМ определяется как $f = \frac{f_{OSC}}{N \cdot (1 + TOP)}$.

Счетный регистр TCNT1 считает от 0 до максимального значения, после чего сбрасывается. Быстрый ШИМ может работать в неинвертирующем и инвертирующем режимах. В неинвертирующем режиме при совпадении значений TCNT1 и одного из регистров OCR1A/OCR1B состояние соответствующего вывода OC1A/OC1B сбрасывается в 0, а при переполнении счетного регистра устанавливается в «1». В инвертирующем режиме работа инверсна. Бит прерывания TOV1 устанавливается каждый раз при переполнении счетчика.

Фазовый ШИМ. Разрядность ШИМ может быть фиксирована (8, 9, 10 бит) или определяться одним из регистров OCR1A или ICR1 и может определяться аналогично быстрому ШИМ. В неинвертирующем режиме выводы OC1A/OC1B устанавливаются в «1» при совпадении TCNT1 и OCR1A/OCR1B при суммирующем счете и сбрасываются в «0» при совпадении TCNT1 и OCR1A/OCR1B при вычитающем счете. В инвертирующем режиме выводы OC1A/OC1B работает инверсно. Флаг прерывания TOV1 устанавливается каждый раз при достижении счетным регистром максимального значения. Частота ШИМ в фазовом режиме определяется как $f = \frac{f_{OSC}}{2 \cdot N \cdot TOP}$.

Фазо-частотный ШИМ аналогичен фазовому. Отличие состоит в моменте перезаписи регистра OCR1x, участвующего в сравнении со счетным регистром. В фазовом режиме перезапись значений регистров OCR1x и модуля пересчета осуществляется в момент достижения максимального значения счетным регистром, а в фазо-частотном – в момент совпадения значений регистров OCR1x и TCNT1.

Предварительный делитель частоты

Таймеры T0 и T1 имеют 10-битные предварительные делители частоты. Сброс предварительного делителя частоты производится единичным значением бита PSR10 регистра SFIOR (см. рис. 14).

8-битный таймер-счетчик T2

Таймер-счетчик T2 имеет возможность работы в следующих режимах:

- измерение промежутков времени;
- ШИМ;
- часы реального времени.

Упрощенная структурная схема таймера T2 представлена на рис. 24.

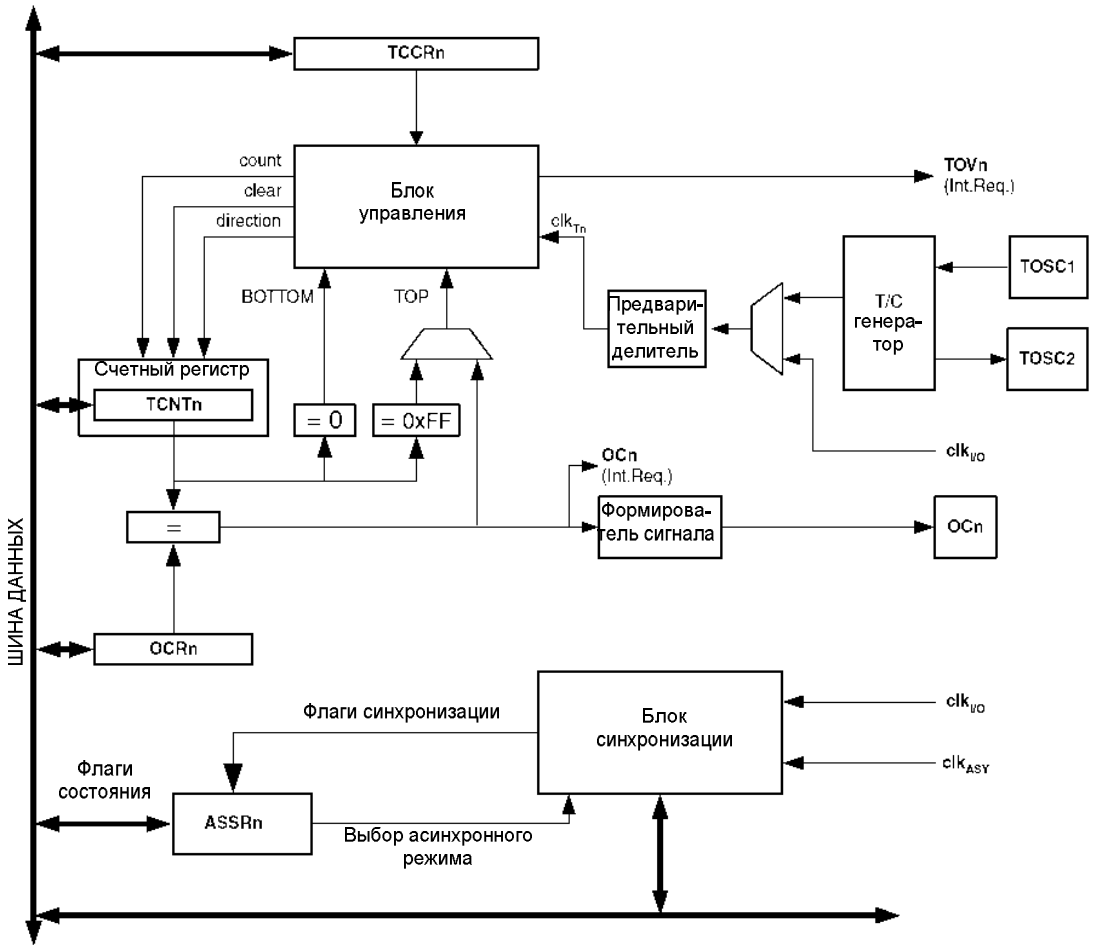


Рис. 24. Упрощенная структурная схема таймера T2 микроконтроллера

Управление таймером-счетчиком T2 осуществляет регистр TCCR2 (рис. 25):

FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
7	6	5	4	3	2	1	0	

Рис. 25. Регистр TCCR2 микроконтроллера

Назначение битов аналогично регистру TCCR0 управления таймером T0.

Отличием таймера T2 от таймера T0 является невозможность подсчета внешних событий (импульсов на выводе INTn). В остальном, режимы работы двух 8-битных таймеров аналогичны.

Особым режимом работы таймера T2 является асинхронный режим. В асинхронном режиме на вход предварительного делителя поступает сигнал от кварцевого генератора таймера-счетчика, что позволяет использовать таймер-счетчик в качестве часов реального времени. В качестве источника сигнала, как правило, используется кварцевый резонатор, подключаемый к выводам TOSC1 и TOSC2 микроконтроллера. Тактовый генератор таймера-счетчика оптимизирован для работы на частоте 32768 Гц, при этом она должна быть как минимум в 4 раза ниже частоты тактового сигнала микроконтроллера.

Для задания асинхронного режима таймера используется регистр ASSR (рис. 26).

-	-	-	-	AS2	TCN2UB	OCR2UB	TCR2UB	ASSR
7	6	5	4	3	2	1	0	

Рис. 26. Регистр ASSR микроконтроллера

AS2 – переключение режима работы. Если бит установлен в «1», то на вход предделителя таймера-счетчика T2 поступают импульсы с кварцевого генератора таймера-счетчика (асинхронный режим). В этом режиме выводы TOSC1 и TOSC2 используются для подключения кварцевого резонатора и соответственно не могут использоваться как контакты ввода-вывода общего назначения. Если бит сброшен в «0», то на вход предделителя поступает внутренний тактовый сигнал микроконтроллера. В этом случае выводы TOSC1 и TOSC2 являются линиями ввода-вывода общего назначения. При изменении состояния этого бита содержимое регистров TCNT2, OCR2 и TCCR2 может быть повреждено.

TCN2UB – ссостояние обновления регистра TCNT2. При записи в регистр TCNT2 этот флаг устанавливается в «1», а после пересылки записываемого значения в данный регистр флаг аппаратно сбрасывается в «0». Таким образом, сброшенный флаг TCN2UB означает, что регистр TCNT2 готов для записи в него нового значения. Запись в регистр TCNT2 при установленном флаге TCN2UB может привести к повреждению прежнего содержимого регистра и к генерации прерывания.

OCR2UB – состояние обновления регистра OCR2. При записи в регистр сравнения соответствующий флаг устанавливается в «1», а после пересылки записываемого значения в регистр флаг аппаратно сбрасывается в «0». Таким образом, сброшенный флаг OCR2UB означает, что соответствующий регистр сравнения готов для записи в него нового значения. Запись в регистр сравнения при установленном флаге OCR2UB может привести к повреждению прежнего содержимого регистра и к генерации прерывания.

TCR2UB – ссостояние обновления регистра TCCR2. При записи в регистр управления соответствующий флаг устанавливается в «1», а после пересылки записываемого значения в регистр флаг аппаратно сбрасывается в «0». Таким образом, сброшенный флаг TCR2UB означает, что соответствующий регистр управления готов для записи в него нового значения. Запись в регистр управления при установленном флаге TCR2UB может привести к повреждению прежнего содержимого регистра и к генерации прерывания.

При переключении между синхронным и асинхронным режимами содержимое регистров таймера-счетчика может быть повреждено. Чтобы этого избежать, рекомендуется придерживаться следующей последовательности действий:

1. Запретить прерывания от таймера-счетчика.
2. Переключить его в требуемый режим.
3. Записать новые значения в регистры TCNT2, OCR2 и TCCR2.
4. В случае переключения в асинхронный режим дождаться сброса флагов TCN2UB, OCR2UB и TCR2UB.
5. Сбросить флаги прерываний таймера-счетчика.
6. Разрешить прерывания (если требуется).

При работе таймера-счетчика в асинхронном режиме установка флагов прерываний от него производится синхронно с тактовым сигналом микроконтроллера. Для синхронизации требуется 3 такта плюс один период тактового сигнала таймера-счетчика. Поэтому к моменту, когда микроконтроллер сможет прочитать состояние счетчика, вызвавшее установку флага прерывания, оно изменится, по меньшей мере, на единицу. Изменение состояния вывода OC2 производится по тактовому сигналу таймера-счетчика и не синхронизируется с тактовым сигналом микроконтроллера.

Сторожевой таймер

Все микроконтроллеры семейства Mega имеют в своем составе сторожевой таймер, предназначенный для защиты микроконтроллера от сбоев в процессе работы (рис. 27).

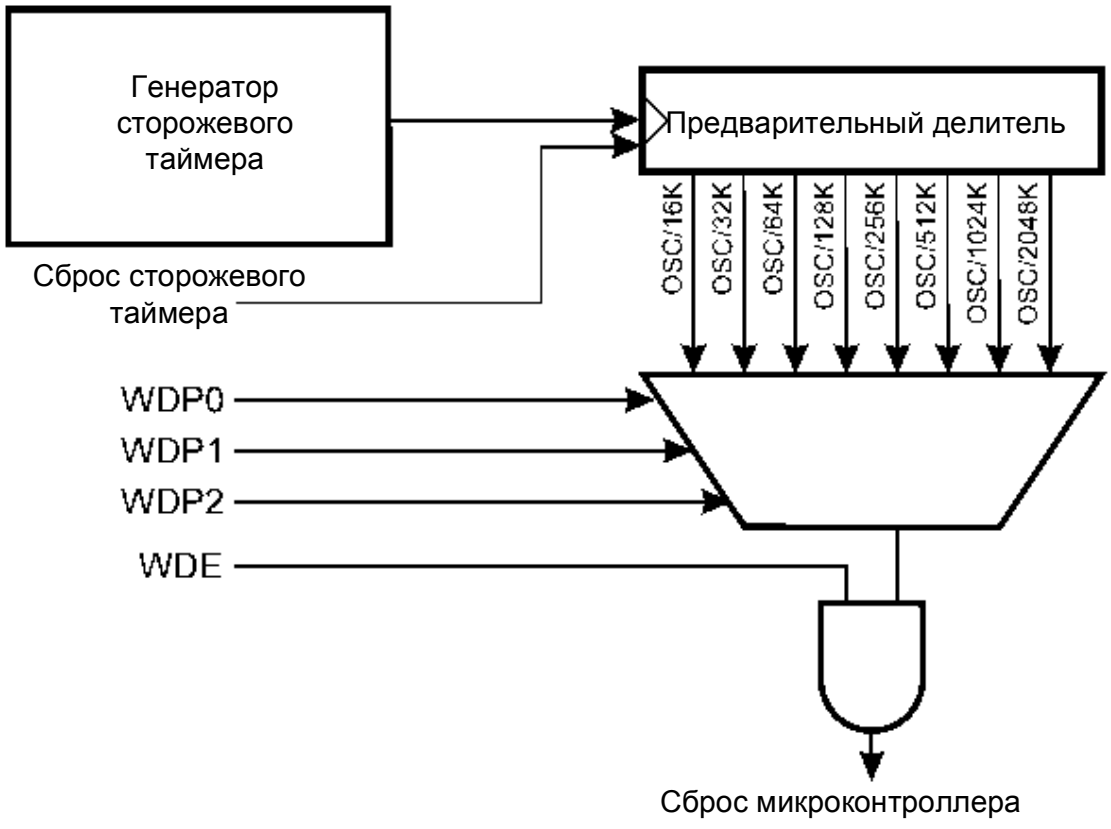


Рис. 27. Структурная схема сторожевого таймера

Сторожевой таймер тактируется отдельным генератором с частотой 1 МГц. Если сторожевой таймер включен, то через промежутки времени, равные его периоду, он выполняет сброс микроконтроллера. Чтобы избежать сброса при нормальном выполнении программы, сторожевой таймер необходимо регулярно сбрасывать через промежутки времени, меньшие его периода. Сброс сторожевого таймера осуществляется командой WDR.

Управление сторожевым таймером осуществляет регистр WDTCR (рис. 28).

-	-	-	WDCE	WDE	WDP2	WDP1	WDP0	WDTCR
7	6	5	4	3	2	1	0	

Рис. 28. Регистр WDTCR микроконтроллера

WDCE – разрешение изменения конфигурации сторожевого таймера: должен быть установлен при записи 0 в WDE для отключения сторожевого таймера, сбрасывается аппаратно через 4 такта.

WDE – разрешение сторожевого таймера: «1» – включен, «0» – выключен.

WDP2:WDP0 – управление предварительным делителем сторожевого таймера

0 0 0 – коэффициент предварительного деления 16К (период WDT 17,1 мс);
0 0 1 – коэффициент предварительного деления 32К (период 34,3 мс);
0 1 0 – коэффициент предварительного деления 64К (период 68,5 мс);
0 1 1 – коэффициент предварительного деления 128К (период 0,14 с);
1 0 0 – коэффициент предварительного деления 256К (период 0,27 с);
1 0 1 – коэффициент предварительного деления 512К (период 0,55 с);
1 1 0 – коэффициент предварительного деления 1024К (период 1,1 с);
1 1 1 – коэффициент предварительного деления 2048К (период 2,2 с).

Для возможности использования сторожевого таймера микроконтроллера на этапе программирования должен быть установлен бит WDTON конфигурационных ячеек (Fuse Bits).

Для отключения сторожевого таймера необходимо выполнить следующие действия:

1. Одной командой записать значения «1» в биты WDCE и WDE регистра WDTCR.
2. В течение следующих 4 тактов записать «0» в бит WDE.

ПРЕРЫВАНИЯ

Прерывание (англ. interrupt) – сигнал, сообщающий процессору о наступлении какого-либо события. При этом выполнение текущей последовательности команд приостанавливается, и управление передается обработчику прерывания, который выполняет работу по обработке события и возвращает управление в прерванный код.

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

Микроконтроллеры AVR семейства Mega имеют многоуровневую систему приоритетных прерываний. Младшие 20 адресов памяти программ, начиная с адреса 0x0001, отведены под таблицу векторов прерываний. Каждому прерыванию соответствует адрес в этой таблице, который загружается в счетчик команд при возникновении прерывания. Положение вектора в таблице также определяет и приоритет соответствующего прерывания: чем меньше адрес, тем выше приоритет прерывания. Размер вектора прерывания зависит от объема памяти программ микроконтроллера и составляет 1 байт.

Положение таблицы векторов прерываний может быть изменено. Таблица может располагаться не только в начале памяти программ, но и в начале области загрузчика, причем перемещение таблицы может быть осуществлено непосредственно в ходе выполнения программы. Для управления размещением таблицы прерываний используется регистр управления микроконтроллера GICR (рис. 29).

INT1	INT0	INT2	-	-	-	IVSEL	IVCE	GICR
7	6	5	4	3	2	1	0	

Рис. 29. Регистр GICR микроконтроллера

INT1 – разрешение внешнего прерывания INT1: если в этом бите записана логическая «1» и флаг I регистра SREG также установлен в «1», то разрешается внешнее прерывание с вывода INT1.

INT0 – разрешение внешнего прерывания INT0: если в этом бите записана логическая «1» и флаг I регистра SREG также установлен в «1», то разрешается внешнее прерывание с вывода INT0.

INT2 – разрешение внешнего прерывания INT2: если в этом бите записана «1» и флаг I регистра SREG установлен в «1», то разрешается внешнее прерывание с вывода INT2.

IVSEL – определяет положение таблицы векторов прерываний в памяти программ. Если флаг сброшен, то таблица векторов прерываний располагается в начале памяти программ, если установлен в «1» – в начале области загрузчика. Конкретное значение начального адреса области загрузчика зависит от установок конфигурационной ячейки BOOTRST.

IVCE – предназначен для разрешения изменения флага IVSEL.

Для изменения положения таблицы векторов прерываний необходимо выполнить следующие действия:

1. Установить бит IVCE в «1».

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

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

Таблица 3

Таблица векторов прерываний микроконтроллера ATmega8535

№	Адрес	Источник	Описание
1	0x0000	RESET	Внешний сброс, сброс при включении питания, сброс монитора напряжения питания, сброс сторожевого таймера
2	0x0001	INT0	Внешнее прерывание 0
3	0x0002	INT1	Внешнее прерывание 1
4	0x0003	TIMER2_COMP	Совпадение таймера/счетчика T2
5	0x0004	TIMER2_OVF	Переполнение таймера/счетчика T2
6	0x0005	TIMER1_CAPT	Захват таймера/счетчика T1
7	0x0006	TIMER1_COMPA	Совпадение A таймера/счетчика T1
8	0x0007	TIMER1_COMPB	Совпадение B таймера/счетчика T1
9	0x0008	TIMER1_OVF	Переполнение таймера/счетчика T1
10	0x0009	TIMER0_OVF	Переполнение таймера/счетчика TO
11	0x000A	SPI, STC	Передача по SPI завершена
12	0x000B	USART, RXC	USART, прием завершен
13	0x000C	USART, UDRE	Регистр данных USART пуст
14	0x000D	USART, TXC	USART, передача завершена
15	0x000E	ADC	Преобразование АЦП завершено
16	0x000F	EE_RDY	EEPROM готово
17	0x0010	ANA_COMP	Аналоговый компаратор
18	0x0011	TW1	Прерывание от модуля TWI
19	0x0012	INT2	Внешнее прерывание 2
20	0x0013	TIMER0_COMP	Совпадение таймера/счетчика TO
21	0x0014	SPM_RDY	Готовность SPM

Для разрешения прерываний должен быть установлен флаг I регистра SREG. Индивидуальное разрешение или запрет прерываний осуществляется установкой или сбросом соответствующих битов регистров масок. При возникновении прерывания флаг I регистра SREG аппаратно сбрасывается, запрещая тем самым обработку следующих прерываний. В программе обработчика прерывания можно снова установить этот флаг для разрешения вложенных прерываний. При возврате из подпрограммы обработки прерывания (при выполнении команды RETI) флаг I устанавливается аппаратно.

При вызове подпрограмм обработки прерываний регистр состояния SREG не сохраняется. Поэтому пользователь должен самостоятельно запоминать содержимое этого регистра при входе в подпрограмму обработки прерывания

(если это необходимо) и восстанавливать его значение перед вызовом команды RETI.

Очередь прерываний работает следующим образом: если условия генерации одного или более прерываний возникают в то время, когда флаг общего разрешения прерываний сброшен (все прерывания запрещены), соответствующие флаги устанавливаются в «1» и остаются в этом состоянии до установки флага общего разрешения прерываний. После разрешения прерываний выполняется их обработка в порядке приоритета.

Наименьшее время отклика для любого прерывания составляет 4 такта. В течение этого времени происходит сохранение счетчика команд в стеке. В течение последующих двух тактов выполняется команда перехода к подпрограмме обработки прерывания. Если прерывание произойдет во время выполнения команды, длящейся несколько циклов, то генерация прерывания произойдет только после выполнения этой команды. Если прерывание произойдет во время нахождения микроконтроллера в «спящем» режиме, то время отклика увеличивается еще на 4 или 5 тактов. Возврат в основную программу занимает 4 такта, в течение которых происходит восстановление счетчика команд из стека. После выхода из прерывания процессор всегда выполняет одну команду основной программы, прежде чем обслужить любое отложенное прерывание.

Внешние прерывания

Внешние прерывания генерируются при изменении состояния на выводах INT0, INT1, INT2 даже в случае, если указанные выводы сконфигурированы как выходы. Для маскирования внешних прерываний используются регистры MCUCR и MCUCSR (рис. 30).

SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
-	ISC2	-	-	WDRF	BORF	EXTRF	PORF	MCUCSR
7	6	5	4	3	2	1	0	

Рис. 30. Регистры MCUCR и MCUCSR микроконтроллера

SM2:SM0 – управление спящим режимом микроконтроллера

0 0 0 – режим холостого хода;
0 0 1 – режим снижения шумов АЦП;
0 1 0 – режим микропотребления;
0 1 1 – экономичный режим;
1 0 0 – зарезервировано;
1 0 1 – зарезервировано;
1 1 0 – режим ожидания;
1 1 1 – расширенный режим ожидания.

SE – разрешение перехода в режим пониженного энергопотребления: установка этого бита в «1» разрешает перевод микроконтроллера в режим пониженного энергопотребления по команде SLEEP, при сброшенном бите SE выполнение команды SLEEP не производит никаких действий.

ISC11:ISC10, ISC01:ISC00 – определяют условия генерации внешних прерываний на выводах INT1/INT0:

0 0 – прерывание при низком уровне на выводе INT1/INT0;
0 1 – прерывание при любом изменении сигнала на выводе INT1/INT0;
1 0 – прерывание по спадающему фронту на выводе INT1/INT0;

1 1 – прерывание по нарастающему фронту на выводе INT1/INT0.

ISC2 – определяет условия генерации внешних прерываний на выводе INT2:

0 – прерывание по спадающему фронту на выводе INT2;

1 – прерывание по нарастающему фронту на выводе INT2.

WDRF –флаг сброса сторожевого таймера: устанавливается в «1» при сбросе сторожевого таймера, сбрасывается при включении питания или программно записью «0».

BORF – флаг сброса монитора питания: устанавливается в «1» при сбросе монитора питания, сбрасывается при включении питания или программно записью «0».

EXTRF – флаг внешнего сброса: устанавливается в «1» при внешнем сбросе, сбрасывается при включении питания или программно записью «0».

PORF – флаг сброса по питанию: устанавливается в «1» при включении питания, сбрасывается только программно записью «0».

Режимы пониженного энергопотребления

Микроконтроллер ATmega8535 поддерживает 6 режимов пониженного энергопотребления. Режимы отличаются числом периферийных устройств микроконтроллера, функционирующих во время «сна» микроконтроллера, и соответственно степенью уменьшения энергопотребления. Для управления режимом пониженного энергопотребления используется регистр MCUCR (см. рис. 31).

Idle (режим холостого хода). В этом режиме прекращается формирование тактовых сигналов для ЦПУ и FLASH-памяти. При этом ЦПУ микроконтроллера останавливается, а все остальные периферийные устройства (интерфейсные модули, таймеры-счетчики, аналоговый компаратор, АЦП, сторожевой таймер), а также подсистема прерываний продолжают функционировать. Поэтому выход из режима Idle возможен как по внешнему, так и по внутреннему прерыванию. Если разрешена работа АЦП, то преобразование начнет выполняться сразу же после перехода в «спящий» режим. Основным преимуществом режима Idle является быстрая реакция на события, приводящие к «пробуждению» микроконтроллера. Выполнение программы начинается сразу же после перехода из режима Idle в рабочий режим.

ADC Noise Reduction (режим снижения шумов АЦП). В этом режиме прекращает работу ЦПУ микроконтроллера и подсистема ввода-вывода (отключаются тактовые сигналы ЦПУ, FLASH, портов ввода-вывода), а АЦП, подсистема обработки внешних прерываний, сторожевой таймер, асинхронный 8-битный таймер-счетчик, блок сравнения адреса модуля TWI и схема обнаружения состояния СТАРТ продолжают функционировать. За счет этого уменьшаются помехи на входах АЦП, вызываемые работой системы ввода-вывода микроконтроллера, что, в свою очередь, позволяет повысить точность преобразования. Если АЦП включен, преобразование начнет выполняться сразу же после перехода в «спящий» режим. Поскольку тактовый сигнал подсистемы ввода-вывода в этом режиме не генерируется, возврат микроконтроллера в рабочий режим может произойти только в результате сброса (аппаратного, от сторожевого таймера, от схемы монитора питания BOD) или в результате генерации следующих прерываний:

- прерывания от сторожевого таймера;
- прерывания от асинхронного таймера-счетчика;
- прерывания по совпадению адреса от интерфейса TWI;
- прерывания по обнаружению состояния СТАРТ;
- внешнего прерывания, обнаруживаемого асинхронно;
- прерывания по изменению состояния выводов;
- прерывания от EEPROM-памяти и SPM-прерывания;
- прерывания от АЦП.

Power Down (режим микропотребления). В данном режиме отключаются все внутренние тактовые сигналы, соответственно прекращается функционирование всех систем микроконтроллера, работающих в синхронном режиме. Единственными узлами, продолжающими работать в этом режиме, являются асинхронные модули микроконтроллера: сторожевой таймер (если он включен), подсистема обработки внешних прерываний, блок сравнения адреса модуля TWI. Выход из режима Power Down возможен либо в результате сброса (аппаратного, от сторожевого таймера, от схемы BOD) или в результате генерации следующих прерываний:

- прерывания по совпадению адреса от интерфейса TWI;
- внешнего прерывания (обнаруживаемого асинхронно).

Поскольку тактовый генератор микроконтроллера в режиме Power Down останавливается, между наступлением события, приводящего к «пробуждению» микроконтроллера, и началом его работы проходит некоторое время, в течение которого тактовый генератор микроконтроллера выходит на рабочий режим. Эта задержка определяется теми же конфигурационными ячейками, которые задают задержку сброса микроконтроллера. Для «пробуждения» микроконтроллера по внешнему прерыванию, генерируемому по низкому уровню, длительность активного сигнала на входе микроконтроллера должна быть не меньше времени запуска микроконтроллера. Если сигнал, вызвавший «пробуждение» микроконтроллера, будет снят раньше, чем микроконтроллер перейдет в рабочий режим, то обработчик соответствующего прерывания вызван не будет!

Power Save (экономичный режим). Режим практически идентичен режиму Power Down, за исключением поведения 8-битного таймера/счетчика, поддерживающего работу в асинхронном режиме. Асинхронный таймер/счетчик может работать в асинхронном режиме во время «сна» микроконтроллера. Выход из режима Power Save возможен в результате событий аналогичных режиму Power Down и по прерываниям от асинхронного таймера/счетчика (прерывания должны быть разрешены).

Standby (режим ожидания). Этот режим рекомендуется задействовать только при использовании в качестве источника тактового сигнала встроенного генератора с внешним резонатором. Режим Standby полностью идентичен режиму Power Down, за исключением того, что тактовый генератор продолжает функционировать. Переход микроконтроллера в рабочий режим происходит гораздо быстрее – за 6 тактов.

Extended Standby (расширенный режим ожидания). Режим полностью идентичен режиму Power Save, за исключением того, что тактовый генератор продолжает функционировать. Поэтому интервал между «пробуждением» микроконтроллера и выходом его в рабочий режим составляет всего 6 тактов.

Различия в функционировании режимов пониженного энергопотребления представлены в таблице 4.

Таблица 4

Функционирование режимов пониженного энергопотребления

	Тактирование			Генераторы			Выход из режима пониженного энергопотребления				
Режим	clk_{IO}	clk_{ADC}	clk_{ASY}	Main clock source	Timer Osc	INT0 INT1 INT2	TWI	Готовность SPM /EEPROM	Timer 2	ADC	I/O
Idle	+	+	+	+	+	+	+	+	+	+	+
ADC Noise Reduction		+	+	+	+	+	+	+	+	+	
Power Down						+	+				
Power Save			+		+	+	+		+		
Standby				+		+	+				
Extended standby			+	+	+	+	+		+		

ТАКТИРОВАНИЕ МИКРОКОНТРОЛЛЕРА

В общем случае устройство синхронизации микроконтроллера имеет вид, представленный на рис. 31:

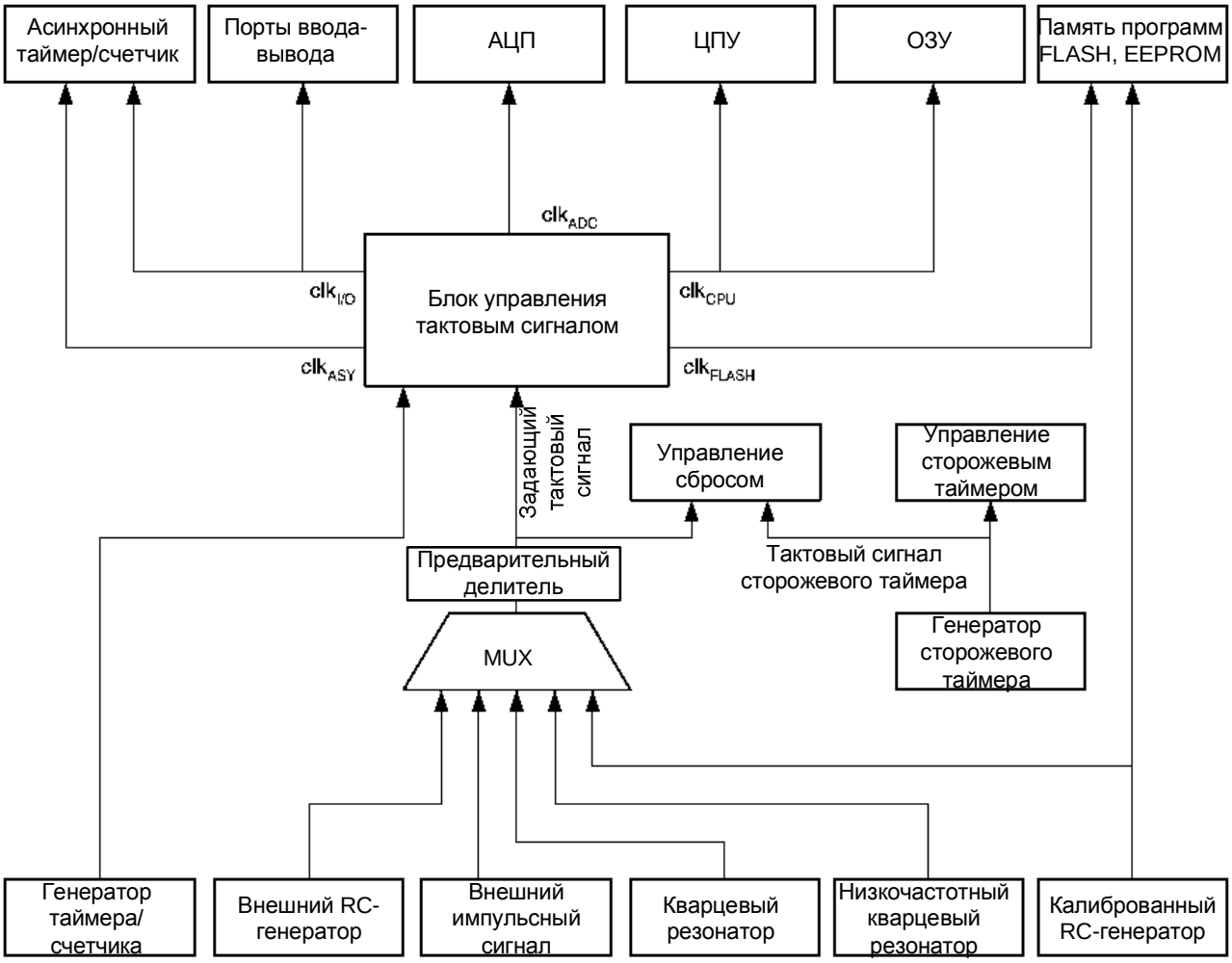


Рис. 31. Устройство синхронизации микроконтроллера

Устройство синхронизации микроконтроллера генерирует следующие сигналы:

- clk_{IO} – тактовый сигнал портов ввода-вывода, используется большинством периферийных устройств, таких как таймеры-счетчики и интерфейсные модули, подсистемой внешних прерываний, однако ряд внешних прерываний может генерироваться и при его отсутствии;
- clk_{FLASH} – тактовый сигнал для FLASH-памяти программ, активируется и деактивируется одновременно с тактовым сигналом центрального процессора clk_{CPU};
- clk_{ASY} – тактовый сигнал асинхронного таймера-счетчика, тактирование осуществляется непосредственно от внешнего кварцевого резонатора 32768 Гц. Наличие данного сигнала позволяет использовать таймер-счетчик T2 в качестве часов реального времени даже при нахождении микроконтроллера в «спящем» режиме;
- clk_{ADC} – тактовый сигнал модуля АЦП, наличие сигнала позволяет осуществлять преобразования при остановленном ЦПУ и подсистеме ввода-

вывода, при этом значительно уменьшается уровень помех, генерируемых микроконтроллером, и соответственно увеличивается точность преобразования.

Тактовый генератор микроконтроллера может работать с внешним кварцевым или керамическим резонатором, внешней или внутренней RC-цепочкой, а также с внешним сигналом синхронизации. Минимально допустимая частота ничем не ограничена (вплоть до пошагового режима работы), а максимальная рабочая частота определяется конкретной моделью микроконтроллера.

Выбор режима работы тактового генератора осуществляется программированием конфигурационных ячеек (FUSE Bits) CKSEL3...0 согласно табл. 5.

Таблица 5

Конфигурирование тактового генератора

Тип генератора	**CKSEL3...0**
Внешний кварцевый резонатор	1111-1010
Внешний низкочастотный кварцевый резонатор	1001
Внешний RC-генератор	1000-0101
Калиброванный внутренний RC-генератор	0100-0001
Внешний импульсный сигнал	0000

По умолчанию используется внутренний RC-генератор (CKSEL3:0=0001).

Генератор с внешним резонатором

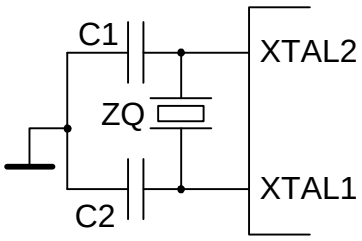


Рис. 32. Подключение кварцевого резонатора

Резонатор подключается к выводам XTAL1 и XTAL2 микроконтроллера (рис. 32). Эти выводы являются соответственно входом и выходом инвертирующего усилителя тактового генератора.

Емкости конденсаторов C1 и C2, подключаемых между выводами резонатора и общим проводом, для кварцевых резонаторов обычно находятся в пределах 12...22пФ.

Усилитель тактового генератора может работать в одном из двух режимов, определяемом состоянием конфигурационного бита CKOPT. Если этот бит запрограммирован (0), то размах колебаний на выходе усилителя (вывод XTAL2) практически равен напряжению питания. Данный режим полезен при работе устройства в условиях сильных электромагнитных помех, а также при использовании сигнала тактового генератора для управления внешними устройствами. В последнем случае между выводом и внешней схемой обязательно должен быть буфер. Если ячейка CKOPT не запрограммирована («1»), то размах колебаний на выходе усилителя будет значительно меньше. Соответственно, ток потребления микроконтроллера уменьшается, однако при этом сужается и диапазон возможных частот тактового сигнала. В этом режиме сигнал тактового генератора нельзя использовать для управления внешними устройствами. Собственно генератор может работать в четырех различных режимах, каждый из которых предназначен для определенного диапазона частот.

Эти режимы определяются ячейками CKSEL3...1 и CKOPT согласно табл. 6.

Таблица 6

Выбор режима генератора с внешним резонатором

CKOPT	CKSEL3...1	Диапазон частот, МГц
1	101	0,4-0,9
1	110	0,9-3,0
1	111	3,0-8,0
0	101, 110, 111	1,0-16,0

Низкочастотный кварцевый генератор

Этот режим предназначен для использования низкочастотного кварцевого резонатора на частоту 32768 Гц, так называемого «часового кварца». Для использования такого режима тактирования выбирается конфигурация битов CKSEL3:0 = 1001. Как и другие внешние резонаторы, он подключается к выводам XTAL1 и XTAL2 микроконтроллеров. Схема подключения низкочастотного кварцевого генератора аналогична схеме для высокочастотного генератора и представлена на рис. 33. При подключении низкочастотного кварцевого генератора можно также использовать внутренние конденсаторы емкостью 36 пФ, которые подключаются при записи 0 в конфигурационный бит CKOPT и позволяют уменьшить количество внешних компонентов схемы.

Внешний сигнал синхронизации

Импульсный сигнал от внешнего источника подается на вывод XTAL1. Этот сигнал должен удовлетворять требованиям микроконтроллера по частоте, скважности и уровням напряжения. Вывод XTAL2 в этом режиме оставляют неподключенным. Между выводом XTAL1 и общим проводом можно включить внутренний конденсатор емкостью 36 пФ записью 0 в конфигурационный бит CKOPT.

Генератор с внешней RC-цепочкой

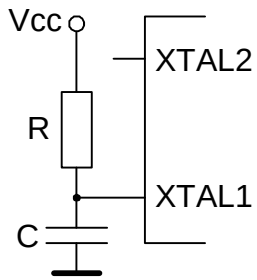


Рис. 33. Подключение внешнего RC-генератора

При реализации приложений, не требующих высокой временной точности, можно использовать простейший RC-генератор. При этом внешняя RC-цепочка подключается к выводу XTAL1 (рис. 33). Емкость конденсатора цепочки должна быть не менее 22 пФ, а сопротивление резистора рекомендуется выбирать из диапазона 3.3... 100 кОм.

Ориентировочно частоту сигнала генератора можно оценить по формуле $f \approx \frac{1}{3 \cdot R \cdot C}$.

Внешний конденсатор можно исключить, задействовав внутренний емкостью 36 пФ, который подключается при записи 0 в конфигурационный бит CKOPT.

При использовании внешней RC-цепочки тактовый генератор может работать в четырех различных режимах, каждый из которых оптимизирован для

определенного диапазона частот. Эти режимы определяются содержимым ячеек CKSEL3...0 и приведены в табл. 7.

Таблица 7

Выбор режима генератора с RC-цепочкой

Внешняя RC-цепочка		Внутренний RC-генератор	
CKSEL3...0	Диапазон частот, МГц	CKSEL3...0	Номинальная частота, МГц
0101	0-0,9	0001	1,0
0110	0,9-3,0	0010	2,0
0111	3,0-8,0	0011	4,0
1000	8,0-12,0	0100	8,0

Внутренний калиброванный RC-генератор

Использование встроенного RC-генератора с внутренней времязадающей RC-цепочкой является наиболее экономичным решением, так как при этом не требуются никакие внешние компоненты. Рабочая частота генератора определяется содержимым конфигурационных битов CKSEL3...0. Возможные конфигурации рабочей частоты приведены в табл. 7.

При работе с внутренним RC-генератором в конфигурационном бите CKOPT должна быть записана «1».

В микроконтроллере предусмотрена возможность подстройки частоты внутреннего генератора (так называемая калибровка). Для этой цели используется регистр OSCCAL, содержащий байт калибровки (табл. 8).

Таблица 8

Калибровка внутреннего RC-генератора

OSCCAL	Минимальная частота, % от номинальной	Максимальная частота, % от номинальной
0x00	50	100
0x7F	75	150
0xFF	100	200

АНАЛОГОВЫЙ КОМПАРАТОР

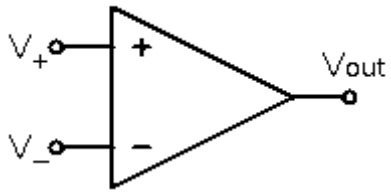


Рис. 34. Аналоговый компаратор

Аналоговый компаратор (рис. 34) – электронная схема, принимающая на свои входы два аналоговых сигнала и выдающая логический «0» или «1», в зависимости от того, какой из сигналов больше. Два входа для подачи аналоговых сигналов носят названия **неинвертирующий (+)** и **инвертирующий (-)**. Если на неинвертирующем входе напряжение больше, чем на инвертирующем, выходной сигнал равен логической «1», иначе – логическому «0».

Будучи включенным, компаратор позволяет сравнивать значения напряжений, присутствующих на двух выводах микроконтроллера:

AIN0/PB2 – неинвертирующий вход;

AIN1/PB3 – инвертирующий вход.

Результатом сравнения является логическое значение, которое может быть прочитано из программы. По результату сравнения может быть сгенерировано прерывание, а также осуществлен захват состояния таймера-счетчика T1. Последняя функция позволяет измерять длительность аналоговых сигналов.

Чтобы указанные выводы можно было использовать компаратором, они должны быть сконфигурированы как входы (соответствующий бит регистра DDRB сброшен в 0). Необходимо также отключить внутренние подтягивающие резисторы путем записи логического «0» в соответствующий бит регистра PORTB.

Структурная схема аналогового компаратора приведена на рис. 35.

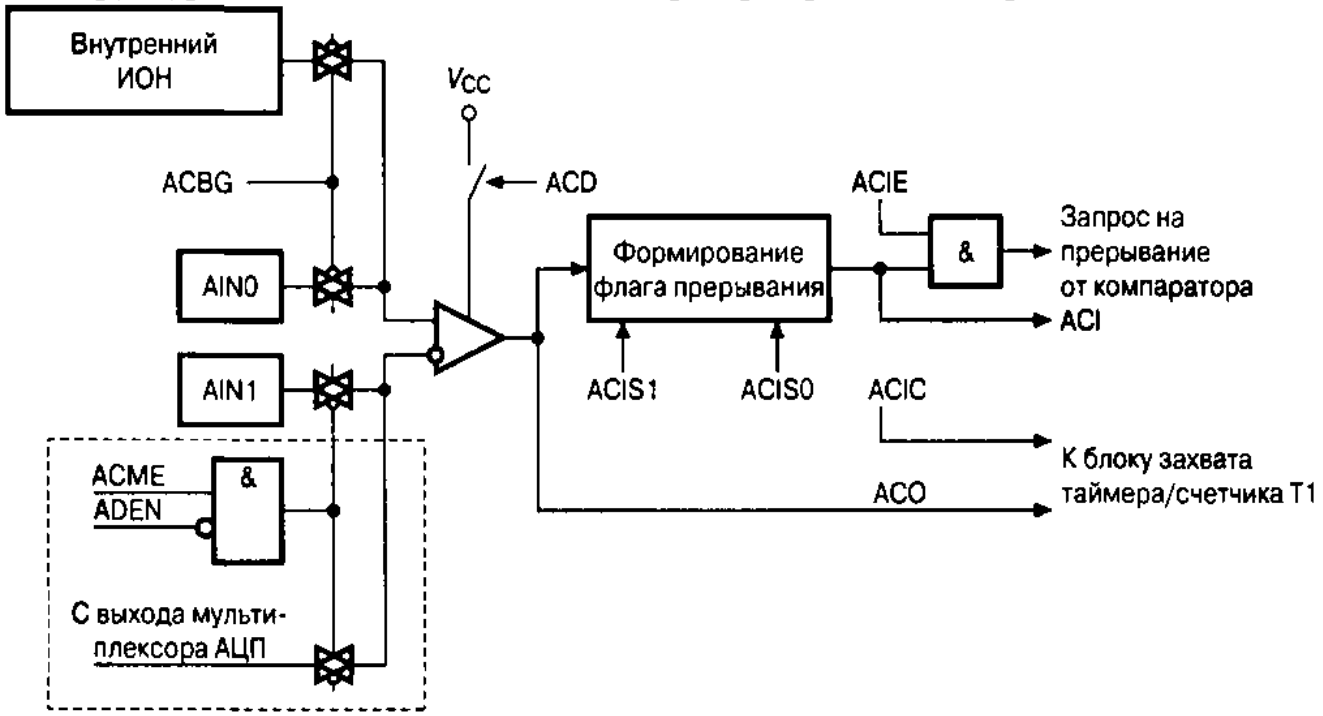


Рис. 35. Структурная схема аналогового компаратора

Основное управление компаратором и контроль его состояния осуществляются с помощью регистра ACSR, представленного на рис. 36.

ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
7	6	5	4	3	2	1	0	

Рис. 36. Регистр ACSR микроконтроллера

ACD – отключение компаратора: «0» – включен, «1» – выключен;

ACBG – подключение к неинвертирующему входу компаратора внутреннего источника опорного напряжения с номинальным напряжением 1,23 В: «0» – не подключен, «1» – подключен;

ACO – результат сравнения (выход компаратора);

ACI – флаг прерывания от компаратора

ACIE – разрешение прерывания от компаратора

ACIC – подключение компаратора к блоку захвата таймера-счетчика T1: «1» – подключен, «0» – отключен;

ACIS1:ACIS0 – условие возникновения прерывания от компаратора

0 0 – прерывание по переключению выхода компаратора;

0 1 – зарезервировано;

1 0 – прерывание по спадающему фронту;

1 1 – прерывание по нарастающему фронту.

Таблица 9

Конфигурация инвертирующего входа компаратора

ACME	ADEN	MUX2:0	Инвертирующий вход
0	x	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

На инвертирующий вход компаратора может поступать сигнал с выхода мультиплексора модуля АЦП. Для этого бит ACME регистра SFIOR необходимо установить в «1» (см. рис. 14). Какой именно из входов АЦП будет использоваться в качестве инвертирующего входа компаратора, определяется битами MUX2...0 регистра ADMUX (рис. 37) микроконтроллера (табл. 9).

REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
7	6	5	4	3	2	1	0	

Рис. 37. Регистр ADMUX микроконтроллера

АНАЛОГО-ЦИФРОВОЙ ПРЕОБРАЗОВАТЕЛЬ

Аналого-цифровой преобразователь (АЦП) – это устройство, преобразующее входной аналоговый сигнал в дискретный код (цифровой сигнал), чаще всего – двоичный. Обратное преобразование осуществляется при помощи цифро-аналогового преобразователя (ЦАП).

В качестве аналогового сигнала может выступать любая физическая непрерывно меняющаяся величина либо ее эквивалент. Часто используют эквивалент напряжения для получения цифровой информации о температуре, токе, частоте и т.д.

Большинство аналого-цифровых преобразователей являются линейными, то есть диапазон входных значений, отображаемый на выходное цифровое значение, связан по линейному закону с этим выходным значением.

Разрешение АЦП – минимальное изменение величины аналогового сигнала, которое может быть преобразовано данным АЦП. Обычно измеряется в вольтах, поскольку для большинства АЦП входным сигналом является электрическое напряжение.

Разрядность АЦП характеризует количество дискретных значений, которые преобразователь может выдать на выходе. Измеряется в битах. Например, АЦП, способный выдать 256 дискретных значений (0..255), имеет разрядность 8 бит.

Разрешение по напряжению равно разности напряжений, соответствующих максимальному и минимальному выходному коду, делённой на количество выходных дискретных значений.

$$d = \frac{U\max - U\min}{2^N},$$

где N – разрядность АЦП.

На практике разрешение АЦП ограничено отношением сигнал-шум входного сигнала. При большой интенсивности шумов на входе АЦП различение соседних уровней входного сигнала становится невозможным, то есть ухудшается разрешение. При этом реально достижимое разрешение описывается **эффективной разрядностью** (effective number of bits — ENOB), которая меньше, чем реальная разрядность АЦП. При преобразовании сильно зашумленного сигнала младшие биты выходного кода практически бесполезны, так как содержат шум.

Дискретизацией сигнала называется измерительное преобразование непрерывного сигнала $x(t)$ в последовательность мгновенных значений этого сигнала $X(k_iT)$, соответствующих определенным моментам времени k_iT (T – шаг дискретизации).

Дискретизацию сигнала по времени можно проводить с постоянным шагом T = const или с переменным шагом T = var (рис. 38).

Частота дискретизации – частота, с которой производится аналого-цифровое преобразование сигнала.

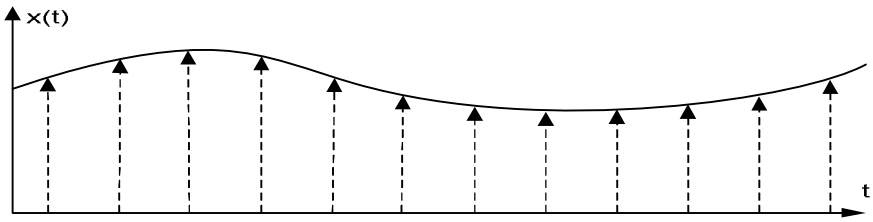


Рис. 38. Дискретизация аналогового сигнала

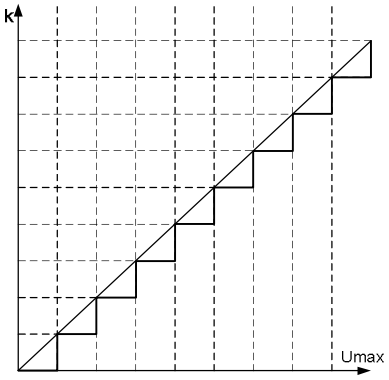


Рис. 39. Идеальная функция преобразования для линейного АЦП

Время преобразования – время от начала преобразования до появления на выходе АЦП соответствующего кода. Время аналого-цифрового преобразования составляет 13-25 машинных циклов.

Опорное напряжение – напряжение, соответствующее максимальному выходному коду.

На рис. 39 представлена идеальная функция преобразования для линейного АЦП (зависимость дискретного отображения аналогового сигнала от его физической величины).

Поскольку реальные АЦП не могут произвести аналого-цифровое преобразование мгновенно, входное аналоговое значение должно удерживаться постоянным, по крайней мере, от начала до конца процесса преобразования (этот интервал времени называют **время преобразования**). Эта задача может решаться путем использования специальной схемы на входе АЦП – устройства выборки-хранения (УВХ). УВХ, как правило, хранит входное напряжение в конденсаторе, который соединен с входом через аналоговый ключ: при замыкании ключа происходит выборка входного сигнала (конденсатор заряжается до входного напряжения), при размыкании – хранение. Многие АЦП, выполненные в виде интегральных микросхем содержат встроенное УВХ.

Основой построения АЦП является аналоговый компаратор.

Существуют следующие типы аналого-цифровых преобразователей:

- АЦП прямого преобразования или параллельный АЦП;
- АЦП последовательного приближения или АЦП с поразрядным уравновешиванием;
- АЦП дифференциального кодирования;
- АЦП сравнения с пилообразным сигналом;
- АЦП с уравновешиванием заряда;
- Конвейерные АЦП;
- Сигма-Дельта АЦП.

Пример осуществления аналого-цифрового преобразования для АЦП с поразрядным уравновешиванием. Входной аналоговый сигнал сравнивается с половинной величиной диапазона преобразования. В случае превышения старший разряд преобразования выставляется в «1», иначе – «0» и коммутируется схема, проверяющая следующий разряд.

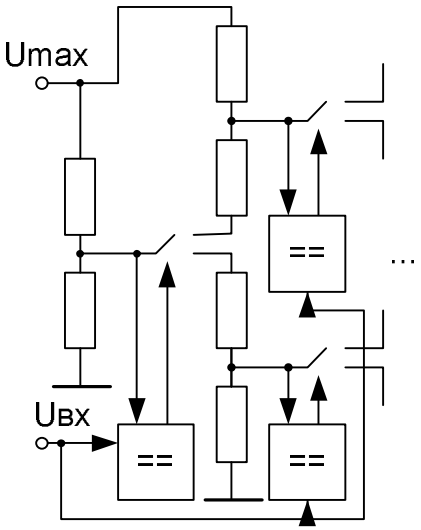


Рис. 40. АЦП с поразрядным уравновешиванием

В состав микроконтроллера ATmega8535 входит модуль 10-битного АЦП последовательного приближения (рис. 40).

На входе модуля АЦП имеется 8-канальный аналоговый мультиплексор, предоставляющий в распоряжение пользователя 8 каналов с несимметричными входами. Входы АЦП могут объединяться попарно для формирования различного числа каналов с дифференциальным входом. В некоторых каналах имеется возможность 10- и 200-кратного предварительного усиления входного сигнала. При коэффициентах усиления 1 и 10 действительная разрешающая способность АЦП по этим каналам составляет 8 бит, а при коэффициенте усиления 200 – 7 бит.

Модуль АЦП может работать в двух режимах:

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

В качестве источника опорного напряжения для АЦП может использоваться как напряжение питания микроконтроллера, так и внутренний либо внешний источник опорного напряжения.

Для простого аналого-цифрового преобразования результирующий код будет определяться по формуле:

$$ADC = \frac{Uin \cdot 1024}{Uref},$$

где *Uin* – преобразуемое аналоговое напряжение; *Uref* – опорное напряжение.

Для дифференциального аналого-цифрового преобразования

$$ADC = \frac{(U^+ - U^-) \cdot GAIN \cdot 512}{Uref}$$

где U^+ – напряжение на неинвертирующем входе компаратора; U^- – напряжение на инвертирующем входе компаратора; *GAIN* – коэффициент усиления; *Uref* – опорное напряжение.

Управление режимами работы АЦП осуществляется с помощью регистров ACSRA (рис. 41а) и ADMUX (рис. 41б).

а)

ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ACSRA
7	6	5	4	3	2	1	0	

б)

REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
7	6	5	4	3	2	1	0	

Рис. 41. Регистры управления режимами работы АЦП микроконтроллера

ADEN – разрешение АЦП: «1» – включено, «0» – выключено;

ADSC – запуск преобразования: 1– начать преобразование

ADATE – выбор режима работы АЦП: «0» – однократное преобразование, «1» – многократное преобразование по нарастающему фронту на выходе триггера;

ADIF – флаг прерывания по окончании АЦП;

ADIE – разрешение прерывания по окончании АЦП;

ADPS2:ADPS0 – выбор частоты преобразования:

0 0 0, 0 0 1 – делитель частоты равен 2;

0 1 0 – делитель частоты равен 4;

0 1 1 – делитель частоты равен 8;

1 0 0 – делитель частоты равен 16;

1 0 1 – делитель частоты равен 32;

1 1 0 – делитель частоты равен 64;

1 1 1 – делитель частоты равен 128.

REFS1:REFS0 – выбор источника опорного напряжения:

0 0 – AREF, внутреннее Vref отключено;

0 1 – AVCC с внешней емкостью на выводе AREF;

1 0 – зарезервировано;

1 1 – внутреннее напряжение Vref=2,56 В с внешней емкостью на AREF.

ADLAR – выравнивание результата преобразования (рис. 43): «0» – слева, «1» – справа.

MUX4:MUX0 – выбор входного канала (табл. 10).

Таблица 10

Выбор входного канала

MUX4..0	Простое АПЦ	Дифф. U+	Дифф. U-	GAIN	MUX4..0	Простое АПЦ	Дифф. U+	Дифф. U-	GAIN
0 0 0 0 0	ADC0				1 0 0 0 0		ADC0	ADC1	1
0 0 0 0 1	ADC1				1 0 0 0 1		ADC1	ADC1	1
0 0 0 1 0	ADC2				1 0 0 1 0		ADC2	ADC1	1
0 0 0 1 1	ADC3				1 0 0 1 1		ADC3	ADC1	1
0 0 1 0 0	ADC4				1 0 1 0 0		ADC4	ADC1	1
0 0 1 0 1	ADC5				1 0 1 0 1		ADC5	ADC1	1
0 0 1 1 0	ADC6				1 0 1 1 0		ADC6	ADC1	1
0 0 1 1 1	ADC7				1 0 1 1 1		ADC7	ADC1	1
0 1 0 0 0		ADC0	ADC0	10	1 1 0 0 0		ADC0	ADC2	1
0 1 0 0 1		ADC1	ADC0	10	1 1 0 0 1		ADC1	ADC2	1
0 1 0 1 0		ADC0	ADC0	200	1 1 0 1 0		ADC2	ADC2	1
0 1 0 1 1		ADC1	ADC0	200	1 1 0 1 1		ADC3	ADC2	1
0 1 1 0 0		ADC2	ADC2	10	1 1 1 0 0		ADC4	ADC2	1
0 1 1 0 1		ADC3	ADC2	10	1 1 1 0 1		ADC5	ADC2	1
0 1 1 1 0		ADC2	ADC2	200	1 1 1 1 0	1.22B			
0 1 1 1 1		ADC3	ADC2	200	1 1 1 1 1	0 B			

Биты ADTS2:ADTS0 регистра SFIOR определяют источник сигнала для запуска многократного аналого-цифрового преобразования (см. рис. 14).

ADTS2:ADTS0 – выбор источника сигнала для запуска многократного АЦП:

0 0 0 – режим непрерывного преобразования;

0 0 1 – прерывание от аналогового компаратора;

0 1 0 – внешнее прерывание INT0;

0 1 1 – прерывание по событию «Совпадение» таймера/счетчика T0;

1 0 0 – прерывание по переполнению таймера/счетчика T0;

1 0 1 – прерывание по событию «Совпадение B» таймера/счетчика T1;

1 1 0 – прерывание по переполнению таймера/счетчика T1;

1 1 1 – прерывание по событию «Захват» таймера/счетчика T1.

На рис. 42 представлена упрощенная структурная схема модуля АЦП.

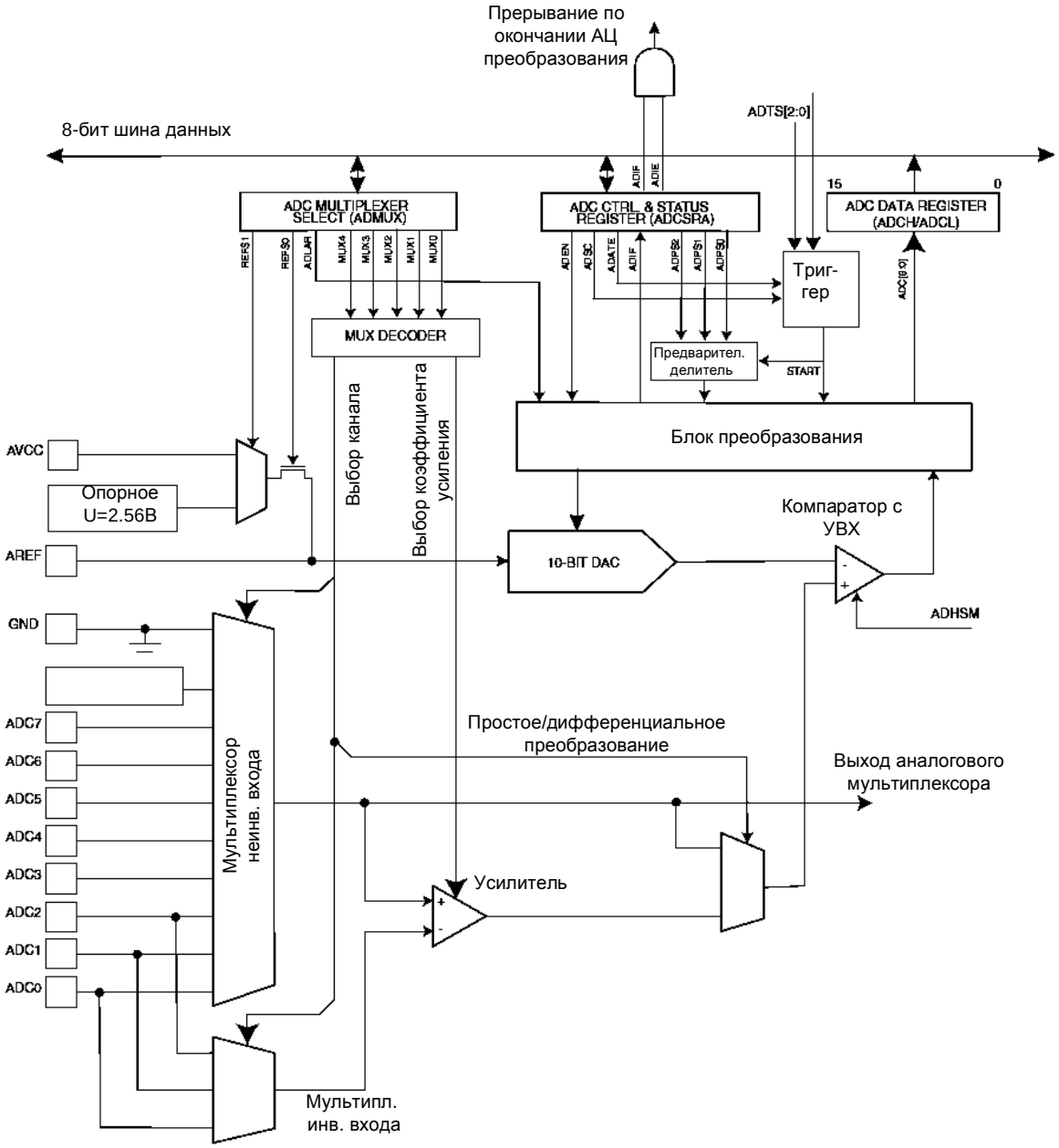


Рис. 42. Структурная схема модуля АЦП

Результат аналого-цифрового преобразования хранится в регистрах ADCH:ADCL и может размещаться двумя способами согласно рис. 43.

ADLAR = 0

-	-	-	-	-	-	ADC9	ADC8	ADCH
ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
7	6	5	4	3	2	1	0	

ADLAR = 1

ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
ADC1	ADC0	-	-	-	-	-	-	ADCL
7	6	5	4	3	2	1	0	

Рис. 43. Размещение результата аналого-цифрового преобразования микроконтроллера

ИНТЕРФЕЙСЫ СВЯЗИ

Интерфейс (interface) – совокупность средств и методов взаимодействия между элементами системы.

Стандарт интерфейса определяет

- механические характеристики интерфейса (разъемы и соединители);
- электрические характеристики сигналов (логические уровни);
- функциональные описания интерфейсных схем (протоколы передачи).

Протокол передачи – стандарт, определяющий поведение функциональных блоков при передаче данных. Большинство протоколов передачи работают по принципу **ведущий – ведомые** (Master – Slave).

Посылка – минимальный объем передаваемых данных, достаточный для интерпретации устройством-приемником в системе связи.

Ведущее устройство (Master) – главное устройство в сети, которое может самостоятельно запрашивать данные у ведомых устройств.

Ведомое устройство (Slave) – устройство, которое не может самостоятельно инициировать передачу своих данных, а передает или принимает их только по запросу ведущего устройства сети.

Большинство интерфейсов используют одно ведущее устройство и одно или несколько ведомых устройств.

Различают дуплексный и полудуплексных режимы работы приемо-передающих устройств. В режиме **дуплекс** устройства могут передавать и принимать информацию одновременно. В режиме **полудуплекс –** или передавать, или принимать информацию.

Скорость передачи информации – скорость передачи данных, выраженная в количестве бит, символов или блоков, передаваемых за единицу времени. Обычно единицей измерения скорости передачи является 1 бод = 1 бит/с.

В микроконтроллере ATmega8535 реализованы:

- последовательный интерфейс SPI:
- универсальный синхронно-асинхронный интерфейс USART;
- двухпроводный интерфейс TWI.

Последовательный интерфейс SPI

SPI (Serial Peripheral Interface) – последовательный синхронный стандарт передачи данных в режиме полного дуплекса, разработанный компанией Motorola для обеспечения простого и недорогого сопряжения микроконтроллеров и периферии. SPI также иногда называют четырехпроводным (англ. four-wire) интерфейсом .

SPI является синхронным протоколом, в котором любая передача синхронизирована с общим тактовым сигналом, генерируемым ведущим устройством (процессором). Принимающая периферия (ведомая) синхронизирует получение битовой последовательности с тактовым сигналом. К одному последовательному периферийному интерфейсу ведущего устройства-микросхемы может присоединяться несколько микросхем. Ведущее устройство

выбирает ведомое для передачи, активируя сигнал «выбор кристалла» (chip select) на ведомой микросхеме. Периферия, не выбранная процессором, не принимает участие в передаче по SPI.

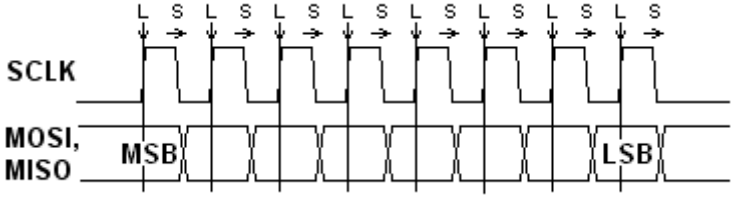
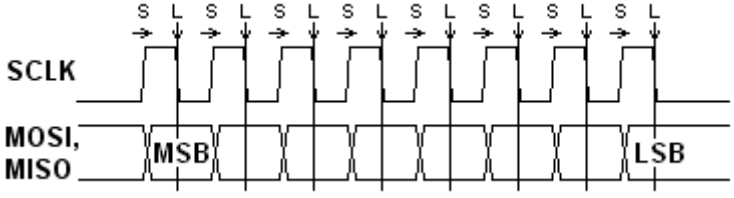
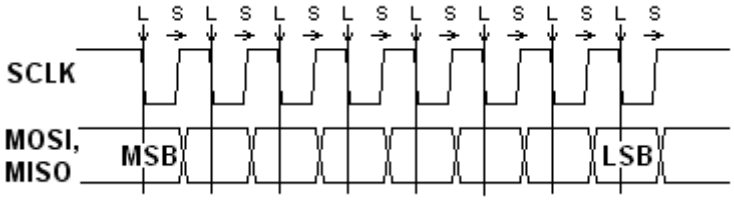
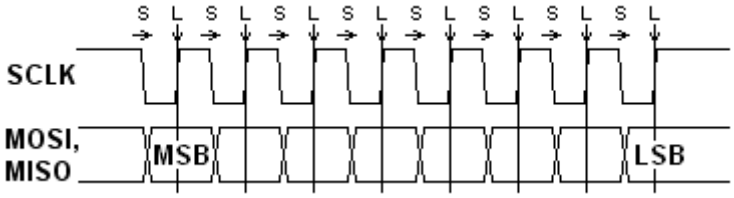
В SPI используются четыре цифровых сигнала:

- MOSI или SI – выход ведущего, вход ведомого (англ. Master Out Slave In). Служит для передачи данных от ведущего устройства ведомому.
- MISO или SO – вход ведущего, выход ведомого (англ. Master In Slave Out). Служит для передачи данных от ведомого устройства ведущему.
- SCLK или SCK – последовательный тактовый сигнал (англ. Serial CLocK). Служит для передачи тактового сигнала для ведомых устройств.
- CS или SS – выбор микросхемы, выбор ведомого (англ. Chip Select, Slave Select). Как правило, выбор микросхемы производится низким логическим уровнем.

В зависимости от комбинаций полярности и фазы синхроимпульсов возможны четыре режима работы SPI, представленные в табл. 11, и мастеру приходится настраиваться на тот режим, который используется ведомым.

Таблица 11

Режимы работы SPI

Режим SPI	Временная диаграмма
Режим "SPI 0" Активные уровень импульсов - высокий. Сначала защёлкивание, затем сдвиг.	
Режим "SPI 1" Активные уровень импульсов - высокий. Сначала сдвиг, затем защёлкивание.	
Режим "SPI 2" Активные уровень импульсов - низкий. Сначала защёлкивание, затем сдвиг.	
Режим "SPI 3" Активные уровень импульсов - низкий. Сначала сдвиг, затем защёлкивание.	

В микроконтроллере интерфейс SPI имеет три назначения:

- с его помощью может осуществляться обмен данными между микроконтроллером и различными периферийными устройствами, такими как цифровые потенциометры, ЦАП/АЦП, FLASH-ПЗУ и др.

- посредством этого интерфейса может производиться обмен данными между несколькими микроконтроллерами AVR;
- через интерфейс SPI может быть осуществлено программирование микроконтроллера (режим последовательного программирования).

При обмене данными по интерфейсу SPI микроконтроллер AVR может работать как ведущий (режим Master) либо как ведомый (режим Slave). При этом пользователь может задавать следующие параметры:

- скорость передачи (восемь программируемых значений);
- формат передачи (от младшего бита к старшему или наоборот).

Дополнительной возможностью подсистемы SPI является «пробуждение» микроконтроллера из режима Idle при поступлении данных.

Структурная схема модуля SPI приведена на рис. 44.

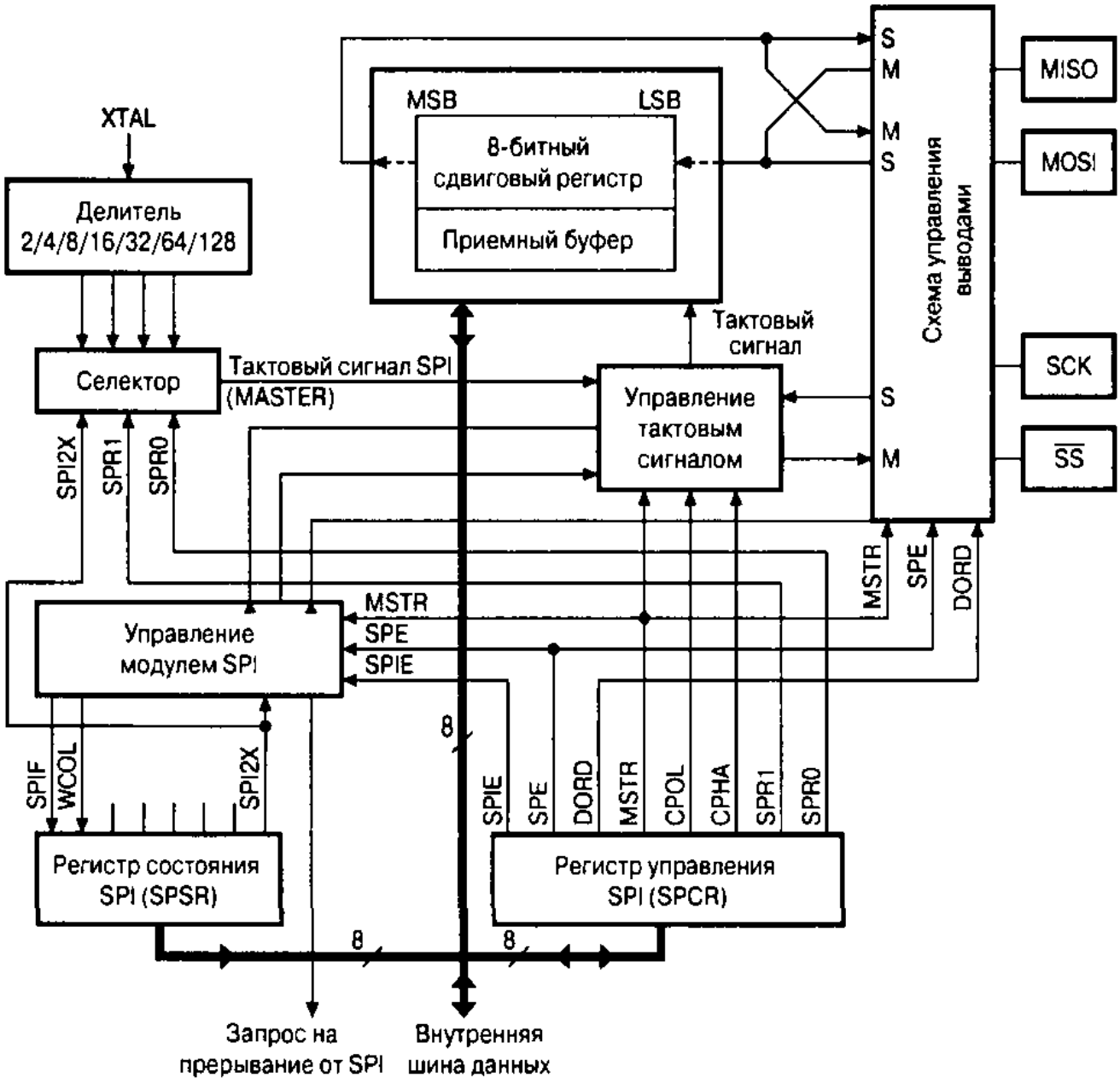


Рис. 44. Структурная схема модуля SPI

Для управления модулем SPI предназначен регистр управления SPCR (рис. 45). Контроль состояния модуля, а также дополнительное управление скоростью обмена осуществляются с помощью регистра состояния SPSR (рис. 45).

SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
SPIF	WCOL	-	-	-	-	-	SPI2X	SPSR
7	6	5	4	3	2	1	0	

Рис. 45. Регистры SPCR, SPSR микроконтроллера

SPIE – разрешение прерывания от SPI: «1» – разрешено, «0» – запрещено.

SPE – включение/выключение SPI: «1» – включен, «0» – выключен.

DORD – порядок передачи данных: «1» – с младшего бита, «0» – со старшего бита.

MSTR – выбор режима работы: «1» – Master, «0» – Slave.

CPOL – полярность тактового сигнала

CPHA – фаза тактового сигнала

CPOL:CPHA

0 0 – SPI 0 1 0 – SPI 2

0 1 – SPI 1 1 1 – SPI 3.

SPR1:SPR0 – скорость передачи ведущего устройства:

при SPI2X=0	при SPI2X=1
0 0 – fosc/4	0 0 – fosc/2
0 1 – fosc/16	0 1 – fosc/8
1 0 – fosc/64	1 0 – fosc/32
1 1 – fosc/128	1 1 – fosc/64

SPIF – флаг прерывания от SPI: устанавливается в «1» по окончании передачи очередного байта. Если бит SPIE регистра SPCR установлен в «1» и прерывания разрешены, то одновременно с установкой флага генерируется прерывание от SPI. Также флаг SPIF устанавливается в «1» при переводе микроконтроллера из режима Master в режим Slave посредством вывода SS. Флаг сбрасывается аппаратно либо при старте подпрограммы обработки прерывания, либо после чтения регистра состояния SPI с последующим обращением к регистру данных SPI (SPDR).

WCOL – флаг конфликта записи: устанавливается в «1» при попытке записи в регистр данных (SPDR) во время передачи очередного байта; сбрасывается аппаратно после чтения регистра состояния SPI с последующим обращением к регистру данных SPI.

SPI2X – удвоение скорости обмена: при установке в «1» и работе микроконтроллера в режиме Master частота сигнала SCK удваивается.

Передаваемые данные записываются, а принимаемые – считываются из регистра данных SPDR. Запись в этот регистр инициирует начало передачи, а при его чтении считывается содержимое буфера сдвигового регистра. Регистр данных служит буфером между регистровым файлом микроконтроллера и сдвиговым регистром модуля SPI.

Соединение двух микроконтроллеров по структуре ведущий – ведомый по интерфейсу SPI показано на рис. 46. Выводы SCK, CS ведущего микроконтроллера является выходами, а ведомого микроконтроллера – входами.

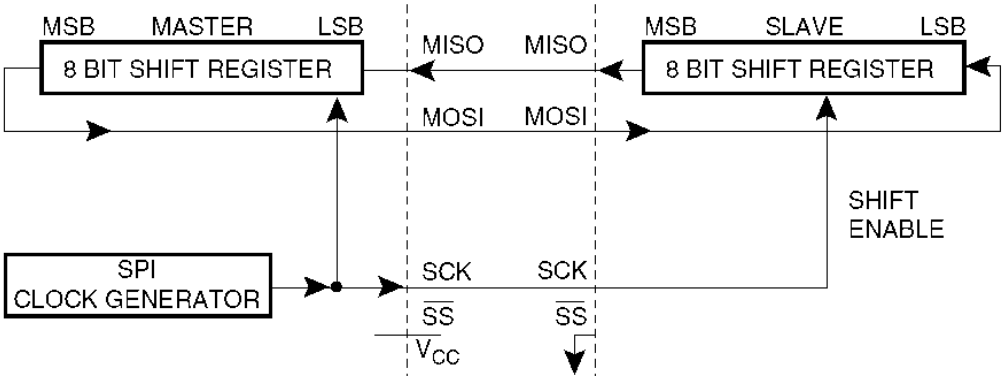


Рис. 46. Соединение двух микроконтроллеров по структуре ведущий – ведомый по интерфейсу SPI

Функции инициализации и обмена данными в режиме SPI (DD_MOSI, DD_MISO и DD_SCK заменить соответствующими выводами):

```
// Инициализация в режиме Master
   void SPI_MasterInit(void)
   {
    DDR = (1<<DD_MOSI)|(1<<DD_SCK);     // MOSI, SCK - выходы
    // включение SPI, Master, частота fck/16
    SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0); //
   }
// Передача данных Ведущим
   void SPI_MasterTransmit(char cData)
   {
    SPDR = cData;         // начало передачи
    while(!(SPSR & (1<<SPIF)));  // ожидание окончания передачи
   }
// Инициализация в режиме Slave
   void SPI_SlaveInit(void)
   {
    DDR_SPI = (1<<DD_MISO);     // MISO - выход
    SPCR = (1<<SPE);            // включение SPI
   }
// Прием данных ведомым
   char SPI_SlaveReceive(void)
   {
    while(!(SPSR & (1<<SPIF)));         // ожидание окончания приема
    return SPDR;                        // передача принятых данных
   }
```

Передача данных осуществляется следующим образом. При записи в регистр данных SPI ведущего микроконтроллера запускается генератор тактового сигнала модуля SPI, и данные начинают побитно выдаваться на вывод MOSI и соответственно поступать на вывод MOSI ведомого микроконтроллера. Порядок передачи битов данных определяется состоянием бита DORD регистра SPCR. После выдачи последнего бита текущего байта генератор тактового сигнала останавливается с одновременной установкой в «1» флага «Конец передачи» (SPIF). Если прерывания от модуля SPI разрешены (флаг SPIE регистра SPCR установлен в «1»), генерируется запрос на прерывание. После этого ведущий микроконтроллер может начать передачу следующего байта либо, подав на вход

SS ведомого напряжение уровня логической «1», перевести его в состояние ожидания.

Одновременно с передачей данных от ведущего к ведомому происходит передача и в обратном направлении, при условии, что на входе SS ведомого присутствует напряжение низкого уровня. Таким образом, в каждом цикле сдвига происходит обмен данными между устройствами. В конце каждого цикла флаг SPIF устанавливается в «1» как в ведущем микроконтроллере, так и в ведомом. Принятые байты сохраняются в приемных буферах для дальнейшего использования.

В модуле реализована одинарная буферизация при передаче и двойная — при приеме. Это означает, что готовый для передачи байт данных не может быть записан в регистр данных SPI до окончания предыдущего цикла обмена. При попытке изменить содержимое регистра данных во время передачи флаг WCOL регистра SPSR устанавливается в «1». Сбрасывается этот флаг после чтения регистра SPSR с последующим обращением к регистру данных SPI.

При приеме данных принятый байт должен быть прочитан из регистра данных SPI до того, как в сдвиговый регистр поступит последний бит следующего байта. В противном случае первый байт будет потерян.

Вывод SS предназначен для выбора активного ведомого устройства и в режиме Slave всегда является входом. Каждый раз, когда на вывод SS подается напряжение уровня логической «1», происходит сброс модуля SPI. Если изменение состояния этого вывода произойдет во время передачи данных, и прием, и передача немедленно прекратятся, а передаваемый и принимаемый байты будут потеряны.

Если микроконтроллер находится в режиме Master (бит MSTR регистра SPCR установлен в «1»), направление передачи данных через вывод SS определяется пользователем. Если вывод сконфигурирован как выход, он работает как линия вывода общего назначения и не влияет на работу модуля SPI. Как правило, в этом случае он используется для управления выводом SS микроконтроллера, работающего в режиме Slave.

Если вывод сконфигурирован как вход, то для обеспечения нормальной работы модуля SPI на него должно быть подано напряжение высокого уровня. Подача на этот вход напряжения низкого уровня от какой-либо внешней схемы будет воспринята модулем SPI как выбор микроконтроллера в качестве ведомого (при этом ему начинают передаваться данные).

Универсальный синхронно-асинхронный приемопередатчик USART

USART – это модуль последовательного ввода-вывода, который может использоваться для работы с периферийными устройствами, такими как терминалы или персональные компьютеры, модемы, микросхемами ЦАП, АЦП, последовательными EEPROM и т.д.

USART может работать в трех режимах:

- асинхронный, полный дуплекс;
- ведущий синхронный, полудуплекс;
- ведомый синхронный, полудуплекс.

Микроконтроллер ATmega8535 имеет в своем составе один модуль универсального синхронно-асинхронного приемо-передатчика USART (Universal Synchronous Asynchronous Receiver Transmitter). Модуль приемо-передатчика обеспечивает полнодуплексный обмен по последовательному каналу, при этом скорость передачи данных может варьироваться в довольно широких пределах. Длина посылки может составлять от 5 до 9 битов. В модуле присутствует схема контроля и формирования бита четности.

Модуль USART, реализованный в микроконтроллере, может обнаруживать следующие внештатные ситуации:

- переполнение;
- ошибка кадрирования;
- неверный старт-бит.

Для уменьшения вероятности сбоев в модуле также реализована функция фильтрации помех. Для взаимодействия с программой предусмотрены три прерывания, запрос на генерацию которых формируется при наступлении следующих событий:

- «передача завершена»;
- «регистр данных передатчика пуст»;
- «прием завершен».

Упрощенная структурная схема модуля USART приведена на рис. 47. Модуль состоит из трех основных частей: блока тактирования, блока передатчика и блока приемника. Блок тактирования модуля USART содержит схему синхронизации, которая используется при работе в синхронном режиме, и контроллер скорости передачи.

Блок передатчика включает одноуровневый буфер, сдвиговый регистр, схему формирования бита четности и схему управления.

Блок приемника содержит схемы восстановления тактового сигнала и данных, схему контроля четности, двухуровневый буфер, сдвиговый регистр, а также схему управления.

Интерфейс USART задействует 3 линии ввода-вывода:

TxD – передача данных;

RxD – прием данных;

XCK – тактовый сигнал (только для синхронного режима).

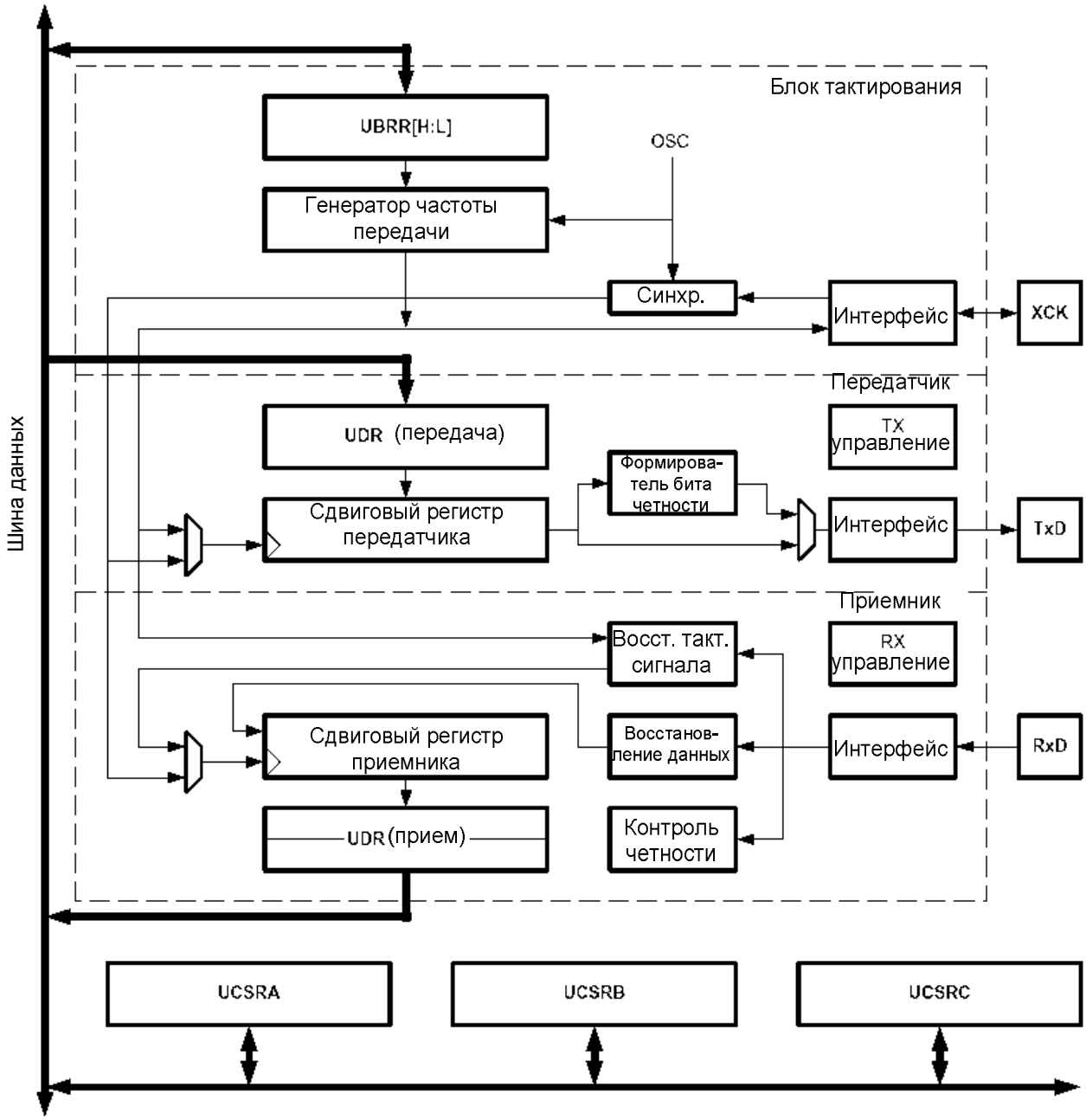


Рис. 47. Упрощенная структурная схема модуля USART

Буферные регистры приемника и передатчика располагаются по одному адресу пространства ввода-вывода и обозначаются как регистр данных UDR. В этом регистре хранятся младшие 8 битов принимаемых и передаваемых данных. При чтении регистра UDR выполняется обращение к буферному регистру приемника, при записи – к буферному регистру передатчика.

Для управления модулем USART используются три регистра: UCSRA, UCSRB и UCSRC (рис. 48).

RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM	UCSRA
RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
URSEL	UMSEL	UPM1	UPM0	USBS	USCZ1	USCZ0	UCPOL	UCSRC
7	6	5	4	3	2	1	0	

Рис. 48. Регистры управления модулем USART

RXC – флаг завершения приема. Флаг устанавливается в «1» при наличии непрочитанных данных в буфере приемника (регистр данных UDR). Сбрасывается флаг аппаратно после опустошения буфера. Если бит RXCIE регистра UCSRB установлен, то при установке флага генерируется запрос на прерывание «прием завершен».

TXC – флаг завершения передачи, устанавливается в «1» после передачи всех битов посылки из сдвигового регистра передатчика при условии, что в регистр данных UDR не было загружено новое значение. Если бит TXCIE регистра UCSRB установлен, то при установке флага генерируется прерывание «передача завершена». Флаг сбрасывается аппаратно при выполнении подпрограммы обработки прерывания или программно, записью в него логического нуля.

UDRE – флаг опустошения регистра данных, устанавливается в «1» при пустом буфере передатчика (после пересылки байта из регистра данных UDR в сдвиговый регистр передатчика). Установленный флаг означает, что в регистр данных можно загружать новое значение. Если бит UDRIE регистра UCSRB установлен, генерируется запрос на прерывание «регистр данных пуст». Флаг сбрасывается аппаратно, при записи в регистр данных.

FE – флаг ошибки кадрирования, устанавливается в «1» при обнаружении ошибки кадрирования, т. е. если первый стоп-бит принятой посылки равен «0», сбрасывается при приеме стоп-бита, равного «1».

DOR – флаг переполнения, устанавливается в «1», если в момент обнаружения нового старт-бита в сдвиговом регистре приемника находится последнее принятое слово, а буфер приемника полон (содержит два байта), сбрасывается при пересылке принятых данных из сдвигового регистра приемника в буфер.

UPE – флаг ошибки контроля четности, устанавливается в «1», если в данных, находящихся в буфере приемника, выявлена ошибка контроля четности. При отключенном контроле четности этот бит постоянно сброшен в «0».

U2X – удвоение скорости обмена, если установлен в «1», то коэффициент деления предделителя контроллера скорости передачи уменьшается с 16 до 8, удваивая тем самым скорость асинхронного обмена по последовательному каналу. Этот бит используется только при асинхронном режиме работы и в синхронном режиме должен быть сброшен.

MPCM – режим мультипроцессорного обмена. Если установлен в «1», ведомый микроконтроллер ожидает приема кадра, содержащего адрес. Кадры, не содержащие адреса устройства, игнорируются.

RXCIE – разрешение прерывания по завершении приема. Если установлен в «1», то при установке флага RXC регистра UCSRA генерируется прерывание «прием завершен» (если флаг I регистра SREG установлен в «1»).

TXCIE – разрешение прерывания по завершении передачи, если установлен в «1», то при установке флага TXC регистра UCSRA генерируется прерывание «передача завершена» (если флаг I регистра SREG установлен в «1»).

UDRIE – разрешение прерывания при очистке регистра данных UART, если установлен в «1», то при установке флага UDRE регистра UCSRA генерируется прерывание «регистр данных пуст» (если флаг I регистра SREG установлен в «1»).

RXEN – разрешение приема, при установке в «1» разрешается работа приемника USART и переопределяется функционирование вывода RXD. При сбросе бита RXEN работа приемника запрещается, а его буфер сбрасывается. Значения флагов TXC, DOR и FE при этом становятся недействительными.

TXEN – разрешение передачи, при установке в «1» разрешается работа передатчика UART и переопределяется функционирование вывода TXD. Если бит сбрасывается в «0» во время передачи, то выключение передатчика произойдет только после завершения передачи данных, находящихся в сдвиговом регистре и буфере передатчика.

UCSZ2:UCSZ0 – используются для задания размера слов данных, передаваемых по последовательному каналу.

0 0 0 – 5 бит
0 0 1 – 6 бит
0 1 0 – 7 бит
0 1 1 – 8 бит
1 0 0 … 1 1 0 – зарезервированы
1 1 1 – 9 бит

RXB8 – 8-й бит принимаемых данных, при использовании 9-битных слов данных содержит значение старшего бита принятого слова. Содержимое этого бита должно быть считано до прочтения регистра данных UDR.

TXB8 – 8-й бит передаваемых данных, при использовании 9-битных слов данных содержимое этого бита является старшим битом передаваемого слова. Требуемое значение должно быть занесено в этот бит до загрузки байта данных в регистр UDR.

URSEL – выбор регистра, определяет, в какой из регистров модуля производится запись. Если бит установлен в «1», обращение производится к регистру UCSRC, если сброшен в «0», обращение производится к регистру UBRRH.

UMSEL – режим работы USART, если сброшен в «0», то модуль работает в асинхронном режиме, если установлен в «1», то модуль работает в синхронном режиме.

UPM1: UPM0 – режим работы схемы контроля и формирования бита четности.

0 0 – контроль четности отключен;
0 1 – зарезервировано;
1 0 – контроль четности включен, проверка на четность (even parity)
1 1 – контроль четности включен, проверка на нечетность (odd parity)

$$P_{EVEN} = d_n \oplus d_{n-1} \oplus d_1 \oplus d_0$$

$$P_{ODD} = d_n \oplus d_{n-1} \oplus d_1 \oplus d_0 \oplus 1$$

USBS – количество стоп-битов, определяет количество стоп-битов, посылаемых передатчиком, если бит сброшен в «0», передатчик посылает 1 стоп-бит, если установлен в «1», то 2 стоп-бита. Для приемника содержимое этого бита безразлично.

UCPOL – полярность тактового сигнала, определяет момент выдачи и считывания данных на выводах модуля, используется только при работе в синхронном режиме (табл. 12). При работе в асинхронном режиме должен быть сброшен.

Таблица 12

Назначение полярности тактового сигнала при работе в синхронном режиме

UCPOL	Выдача данных TxR	Считывание данных RxD
0	спадающий фронт	нарастающий фронт
1	нарастающий фронт	спадающий фронт

В асинхронном режиме, а также в синхронном режиме при работе в качестве ведущего скорость приема и передачи данных задается контроллером скорости передачи, работающим как делитель системного тактового сигнала с программируемым коэффициентом деления. Коэффициент определяется содержимым регистра контроллера UBRR (рис. 49). В блок приемника сформированный сигнал поступает напрямую, а в блок передатчика – через дополнительный делитель, коэффициент деления которого (8 или 16) зависит от режима работы модуля USART.

Регистр UBRR является 12-битным и физически размещается в двух регистрах ввода-вывода UBRRH:UBRRL.

URSEL	-	-	-	UBRR11	UBRR10	UBRR9	UBRR8	UBRRH
UBRR7	UBRR6	UBRR5	UBRR4	UBRR3	UBRR2	UBRR1	UBRR0	UBRRL
7	6	5	4	3	2	1	0	

Рис. 49. Регистры задания скорости передачи по USART интерфейсу

Регистр UBRRH размещается по тому же адресу, что и регистр управления UCSRC. Поэтому при обращении по этому адресу необходимо выполнить ряд дополнительных действий для выбора конкретного регистра.

При записи регистр определяется состоянием старшего бита записываемого значения URSEL. Если этот бит сброшен в 0, изменяется содержимое регистра UBRRH. Если же старший бит значения установлен в «1», изменяется содержимое регистра управления UCSRC.

```
UBRRH = 0x02; // Записать 2 в UBRRH
// Установить биты USBS и UCSZ1 регистра UCSRC
UCSRC = (1<<URSEL) | (1<<USBS) | (1<<UCSZ1);
```

Для выбора регистра при чтении используется временная последовательность. При первом обращении по указанным адресам возвращается значение регистра UBRRH. При повторном обращении по этим адресам в следующем такте возвращается значение регистра UCSRC. Прерывания при выполнении этой последовательности команд должны быть запрещены.

```
unsigned char USART_ReadUCSRC(void)
{
 unsigned char ucsrc;
 ucsrc = UBRRH;
 ucsrc = UCSRC;
 return ucsrc;
}
```

При работе в асинхронном режиме скорость обмена определяется не только содержимым регистра UBRR, но и состоянием бита U2X регистра UCSRA. Если этот бит установлен в «1», коэффициент деления предделителя уменьшается в два раза, а скорость обмена соответственно удваивается. При работе в синхронном режиме этот бит должен быть сброшен.

Итак, скорость обмена определяется следующим образом:

• асинхронный режим (обычный, U2X = 0) $BAUD = \frac{f_{CK}}{16 \cdot (UBRR + 1)}$;

• асинхронный режим (ускоренный, U2X = 1) $BAUD = \frac{f_{CK}}{8 \cdot (UBRR + 1)}$;

• синхронный режим ведущего $BAUD = \frac{f_{CK}}{2 \cdot (UBRR + 1)}$,

где $BAUD$ – скорость передачи, бит/с; f_{CK} – тактовая частота микроконтроллера;

Регистр UBRR содержит значение, соответствующее конфигурации скорости передачи в диапазоне 0…4095.

В некоторых случаях значение регистра UBRR не может точно обеспечить требуемую скорость. При этом возникает ошибка скорости передачи, определяемая как

$$Error[\%] = \frac{BaudRate - BAUD}{BAUD} \cdot 100\%,$$

где $BAUD$ – требуемая скорость передачи, бит/с; $BaudRate$ – расчетная скорость передачи, бит/с.

При использовании скоростей, дающих ошибку больше 0,5 %, снижается помехозащищенность линии передачи.

При работе в синхронном режиме в качестве ведомого скорость приема и передачи определяется частотой сигнала, поступающего на вывод XCK. Частота этого сигнала должна удовлетворять условию

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

Это ограничение связано с тем, что сигнал с вывода XCK сначала синхронизируется с тактовой частотой микроконтроллера, а затем проходит через детектор фронтов. Задержка сигнала при прохождении этих узлов равна двум тактам.

Кадр – совокупность одного слова данных и сопутствующей информации. Кадр начинается со старт-бита, за которым следует младший бит слова данных. После старшего бита слова данных следует один или два стоп-бита. Если включена схема формирования бита четности, он включается между старшим битом слова данных и первым стоп-битом.

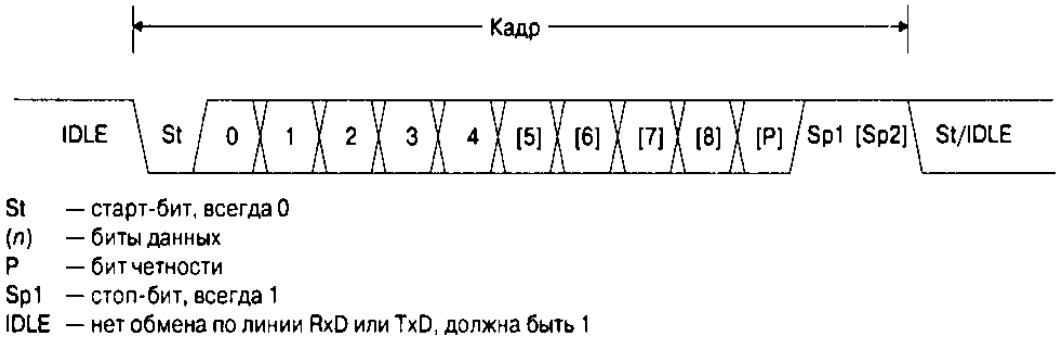


Рис. 50. Кадр команды

Инициализация USART производится функцией

```
void USART_Init( unsigned int baud )
{
 UBRRH = (unsigned char)(baud>>8); // Установка скорости обмена
 UBRRL = (unsigned char)baud;
 UCSRB = (1<<RXEN)|(1<<TXEN); // Разрешение приема и передачи
 // Установка формата кадра: 8бит данных, 2стоп-бита
 UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
```

Передача данных

Работа передатчика разрешается установкой в «1» бита TXEN регистра UCSRB. При установке бита вывод TxD подключается к передатчику USART и начинает функционировать как выход независимо от установок регистров управления портом. Если используется синхронный режим работы, то переопределяется также функционирование вывода XCK. Передача инициируется записью передаваемых данных в буферный регистр передатчика – регистр данных UDR. После этого данные пересылаются из регистра UDR в сдвиговый регистр передатчика. Одновременно, если используются 9-битные данные, значение бита

TXB8 регистра UCSRB копируется в 9-й бит сдвигового регистра. При этом возможны два варианта:

- запись в регистр UDR осуществляется в тот момент, когда передатчик находится в состоянии ожидания (предыдущие данные уже переданы). В этом случае данные пересылаются в сдвиговый регистр сразу же после записи в регистр UDR;
- запись в регистр UDR осуществляется во время передачи. В этом случае данные пересылаются в сдвиговый регистр после передачи последнего стоп-бита текущего кадра.

9-й бит данных должен быть загружен в бит TXB8 до записи младшего байта слова в регистр данных.

После пересылки слова данных в сдвиговый регистр флаг UDRE регистра UCSRA устанавливается в «1», что означает готовность передатчика к получению нового слова данных. В этом состоянии флаг остается до следующей записи в буфер. Одновременно с пересылкой в регистре формируется служебная информация – старт-бит, возможный бит четности, а также один или два стоп-бита.

После загрузки сдвигового регистра его содержимое начинает сдвигаться вправо и поступать на вывод TxD. Скорость сдвига определяется настройками контроллера тактовых сигналов. В синхронном режиме изменение состояния вывода TxD происходит по одному из фронтов сигнала XCK. Если бит UCPOL регистра UCSRC сброшен в 0, изменение состояния вывода происходит по нарастающему фронту сигнала, если установлен в «1» – по спадающему фронту.

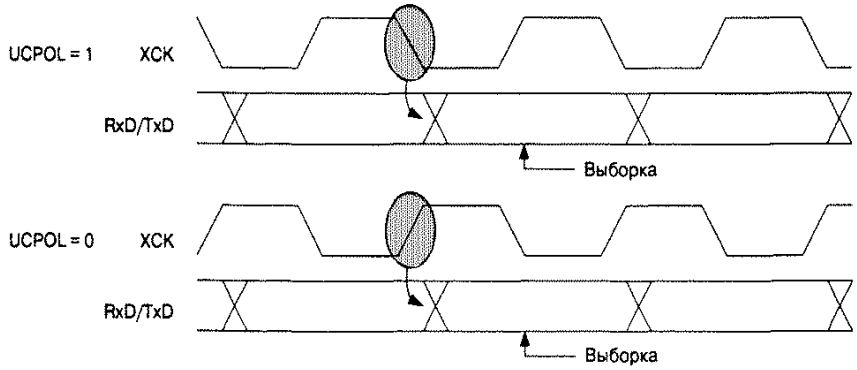


Рис. 51. Временные диаграммы для синхронного режима работы USART

Если во время передачи в регистр UDR было записано новое слово данных, то после передачи последнего стоп-бита оно автоматически пересылается в сдвиговый регистр. Если к моменту окончания передачи кадра буфер передатчика будет пуст, устанавливается флаг прерывания «передача завершена» TXC регистра UCSRA. Сброс флага осуществляется аппаратно при входе в подпрограмму обработки соответствующего прерывания или программно, записью в этот бит логической «1».

Выключение передатчика осуществляется сбросом бита TXEN регистра UCSRB. Если в момент выполнения этой команды осуществлялась передача, сброс бита произойдет только после завершения текущей и отложенной передач,

т. е. после очистки сдвигового и буферного регистров передатчика. При выключенном передатчике вывод TxD может использоваться как контакт ввода-вывода общего назначения.

Ниже приведен простейший пример подпрограммы передачи по интерфейсу USART. Эта подпрограмма ждет очистки буфера передатчика, а затем загружает в него новое значение.

```
void USART_Transmit( unsigned char data )
{
 while (!( UCSRA & (1<<UDRE)));// Ждать очистки буфера передатчика
 UDR = data; // Загрузить данные в буфер для передачи
}
```

Прием данных

Работа приемника разрешается установкой бита RXEN регистра UCSR. При установке бита вывод RxD подключается к приемнику USART и начинает функционировать как вход независимо от установок регистров управления портом. Если используется синхронный режим работы, переопределяется также функционирование вывода XCK.

Прием данных начинается сразу же после обнаружения приемником корректного старт-бита. Каждый бит содержимого кадра затем считывается с частотой, определяемой установками контроллера скорости передачи или тактовым сигналом XCK. Считанные биты данных последовательно помещаются в сдвиговый регистр приемника до обнаружения первого стоп-бита кадра. После этого содержимое сдвигового регистра пересылается в буфер приемника, из которого принятое значение может быть получено путем чтения регистра данных модуля. При использовании 9-битных слов данных значение старшего бита может быть определено по состоянию флага RX8 регистра UCSRB. Содержимое старшего бита данных должно быть считано до обращения к регистру данных. Это связано с тем, что флаг RX8 отображает значение старшего бита слова данных кадра, находящегося на верхнем уровне буфера приемника, состояние которого при чтении регистра данных изменится.

Если во время приема кадра была включена схема контроля четности, она вычисляет бит четности для всех битов принятого слова данных и сравнивает его с принятым битом четности. Результат проверки запоминается в буфере приемника вместе с принятым словом данных и стоп-битами. Наличие или отсутствие ошибки контроля четности может быть затем определено по состоянию флага UPE. Этот флаг устанавливается в «1», если следующее слово, которое может быть прочитано из буфера, имеет ошибку контроля четности. При выключенном контроле четности флаг UPE всегда читается как «0».

Блок приемника модулей USART имеет еще два флага, показывающих состояние обмена, – флаг ошибки кадрирования FE и флаг переполнения DOR. Флаг FE устанавливается в «1», если значение первого стоп-бита принятого кадра не соответствует требуемому, т. е. равно «0». Флаг DOR индицирует потерю данных из-за переполнения буфера приемника. Этот флаг устанавливается в «1» в случае приема старт-бита нового кадра при заполненных буфере и сдвиговом регистре приемника. Установленный флаг DOR означает, что между прошлым

байтом, считанным из регистра UDR, и байтом, считанным в данный момент, произошла потеря одного или нескольких кадров.

Все флаги ошибок буферизуются вместе со словом данных, т.е. соответствующие биты регистра UCSRA относятся к кадру, слово данных которого будет прочитано при следующем обращении к регистру данных UDR. Поэтому состояние этих флагов должно быть считано перед обращением к регистру данных. Для совместимости с будущими устройствами рекомендуется при записи в регистр UCSRA сбрасывать соответствующие этим флагам биты записываемого значения в «0».

Для индикации состояния приемника в модулях USART используется флаг прерывания «прием завершен» RXC регистра UCSRA. Этот флаг устанавливается в «1» при наличии в буфере приемника непрочитанных данных и сбрасывается в «0» при опустошении буфера (после считывания всех находящихся в нем данных).

Выключение приемника осуществляется сбросом бита RXEN регистра UCSRB. В отличие от передатчика, приемник выключается сразу же после сброса бита, а значит, кадр, принимаемый в этот момент, теряется. При выключении приемника очищается его буфер, т. е. теряются также все непрочитанные данные. При выключенном приемнике вывод RxD может использоваться как контакт ввода-вывода общего назначения.

Пример подпрограммы приема по интерфейсу USART.

```
unsigned char USART_Receive( void )
{
 while ( !(UCSRA & (1<<RXC)));// Ждать заполнения буфера приемника
 return UDR;                  // вернуть принятую посылку
}
```

Прием всех битов кадра осуществляется по-разному, в зависимости от режима работы модуля. При работе модуля USART в синхронном режиме состояние вывода RxD считывается по одному из фронтов сигнала XCK. Если бит UCPOL регистра UCSRC сброшен в «0», считывание состояния вывода происходит по спадающему фронту сигнала XCK, если установлен в «1» – по нарастающему фронту сигнала. Считывание данных с вывода RxD и их выдача на вывод TxD происходят по противоположным фронтам.

Для обеспечения приема в асинхронном режиме работы используются схемы восстановления тактового сигнала и данных. Схема восстановления тактового сигнала предназначена для синхронизации внутреннего тактового сигнала, формируемого контроллером скорости передачи, и кадров, поступающих на вывод RxD микроконтроллера. Схема восстановления данных осуществляет считывание и фильтрацию каждого бита принимаемого кадра. Схема восстановления тактового сигнала осуществляет опрос входа приемника с целью определения старт-бита кадра. Частота опроса зависит от состояния бита U2X регистра UCSRA. В обычном режиме (при U2X = 0) частота опроса в 16 раз превышает скорость передачи данных, а в ускоренном режиме (при U2X = 1) – в 8 раз.

Обнаружение спадающего фронта на выводе RxD интерпретируется как возможное появление переднего фронта старт-бита. После этого в нормальном режиме проверяется значение 8-й, 9-й и 10-й выборок входного сигнала, а в ускоренном режиме – 4-й, 5-й и 6-й выборок. Если значение хотя бы двух выборок из указанных равно «1», старт-бит считается ложным (помеха), а приемник переходит к ожиданию следующего изменения входного сигнала с «1» в «0». В противном случае считается, что обнаружен старт-бит новой последовательности, с которым синхронизируется внутренний тактовый сигнал приемника. После этого начинает работать схема восстановления данных.

Решение о значении принятого бита принимается также по результатам 8-й, 9-й и 10-й (4-й, 5-й и 6-й) выборок входного сигнала.

Состоянием бита считается логическое значение, которое было получено, хотя бы в двух из трех выборок. Процесс распознавания повторяется для всех битов принимаемого кадра, включая первый стоп-бит.

Старт-бит нового кадра может передаваться сразу же после последней выборки, используемой для определения значения бита. В обычном режиме работы формирование старт-бита может начаться в момент A, а в ускоренном режиме – в момент B (рис. 52в). Момент C, обозначенный на рисунке, определяет максимальную длительность стоп-бита.

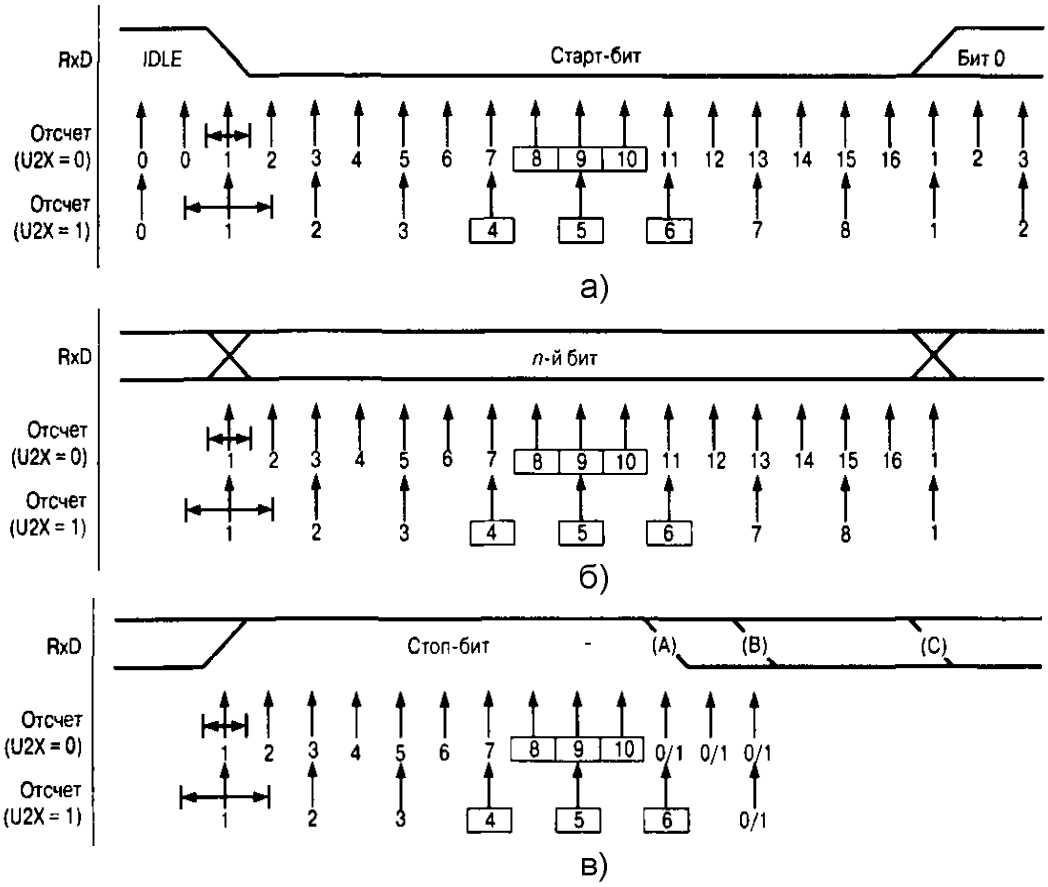


Рис. 52. Распознавание битов кадра: а – старт-бит; б – остальные биты в – стоп-бит

Мультипроцессорный режим работы

Режим мультипроцессорного обмена позволяет осуществлять связь между несколькими ведомыми микроконтроллерами и одним ведущим. В этом режиме каждый ведомый микроконтроллер имеет свой уникальный адрес. Если ведущий микроконтроллер хочет что-либо передать, он посылает адресный байт, определяющий, к какому из микроконтроллеров ведущий собирается обратиться. Если какой-либо из ведомых микроконтроллеров распознал свой адрес, он переходит в режим приема данных и соответственно принимает последующие байты как данные. Остальные ведомые микроконтроллеры игнорируют принимаемые байты до посылки ведущим нового адресного байта. Включение режима фильтрации принимаемых кадров, не содержащих адреса, осуществляется установкой в «1» бита MPCM регистра UCSRA.

В микроконтроллере, выполняющем роль ведущего, должен быть установлен режим передачи 9-битных данных. При передаче адресного байта старший бит должен устанавливаться в «1», а при передаче байтов данных он должен сбрасываться в «0».

В ведомых микроконтроллерах механизм приема зависит от режима работы приемника. Если приемник настроен на прием 5...8-битных данных, то идентификация содержимого кадра (адрес, данные) осуществляется по первому стоп-биту. При приеме 9-битных данных идентификация содержимого кадра осуществляется по старшему биту слова данных. Если указанные биты установлены в «1», значит, кадр содержит адрес, в противном случае – данные.

Для осуществления обмена данными в мультипроцессорном режиме необходимо действовать следующим образом:

1. Все ведомые микроконтроллеры переключаются в режим мультипроцессорного обмена установкой в «1» бита MPCM регистра UCSRA.
2. Ведущий микроконтроллер посылает адресный кадр, а все ведомые микроконтроллеры его принимают. Соответственно в каждом из ведомых микроконтроллеров устанавливается флаг RXC регистра UCSRA.
3. Каждый из ведомых микроконтроллеров считывает содержимое регистра данных. Микроконтроллер, адрес которого совпал с адресом, посланным ведущим, сбрасывает в «0» бит MPCM.

Адресованный микроконтроллер начинает принимать кадры, содержащие данные. Если приемник ведомого микроконтроллера настроен на прием 5...8-битных данных, будет генерироваться ошибка кадрирования, поскольку первый стоп-бит будет равен «0» (этого можно избежать, используя два стоп-бита). В остальных ведомых микроконтроллерах бит MPCM установлен в «1», поэтому кадры данных будут игнорироваться.

После приема последнего байта данных адресованный микроконтроллер устанавливает бит MPCM в «1» и снова ожидает приход кадра с адресом. Процесс повторяется с п. 2.

ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРОВ AVR СЕМЕЙСТВА MEGA

Микроконтроллеры семейства Mega (в частности, ATmega8535) поддерживают следующие режимы программирования:

- режим последовательного программирования (по интерфейсу SPI);
- режим параллельного программирования при высоком напряжении;
- режим самопрограммирования – изменение содержимого памяти программ, управляемое самим микроконтроллером.

Некоторые микроконтроллеры семейства Mega поддерживают режим программирования через интерфейс JTAG.

Под «высоким» напряжением понимается управляющее напряжение 12 В, подаваемое на вывод RESET микроконтроллера для перевода последнего в режим программирования.

В процессе программирования могут выполняться следующие операции:

- стирание кристалла (chip erase);
- чтение/запись FLASH-памяти программ;
- чтение/запись EEPROM-памяти данных;
- чтение/запись конфигурационных ячеек;
- чтение/запись ячеек защиты;
- чтение ячеек идентификатора;
- чтение калибровочного байта.

В исходном состоянии (в новом микроконтроллере) во всех ячейках памяти данных и программ находятся логические «1»: чтение любого байта дает 0xFF. При этом микроконтроллер готов к немедленному программированию.

Защита кода и данных

Для защиты кода и данных используется байт защиты (Lock Bit Byte) (рис. 53):

-	-	BLB12	BLB11	BLB02	BLB01	LB1	LB0	Lock Bit Byte
7	6	5	4	3	2	1	0	

Рис. 53. Байт защиты кода и данных

В исходном состоянии все биты указанного байта равны «1». При программировании в соответствующую ячейку записывается логический «0». Стирание ячеек (запись в них «1») может быть произведено только при выполнении команды «Стирание кристалла», уничтожающей также содержимое FLASH и EEPROM памяти.

BLB12:BLB11 – определяют уровень доступа из секции прикладной программы к коду, расположенному в секции загрузчика.

1 1 – нет никаких ограничений по доступу к коду, расположенному в секции загрузчика;

1 0 – команда SPM не может осуществлять запись по адресам, находящимся в пределах секции загрузчика;

0 1 – команда SPM не может осуществлять запись по адресам, находящимся в пределах секции загрузчика, и команда LPM, вызываемая из секции прикладной программы,

не может осуществлять чтение из секции загрузчика; Если таблица векторов прерываний расположена в секции прикладной программы, то при выполнении кода из секции загрузчика прерывания запрещены;

0 0 – команда LPM, вызываемая из секции прикладной программы, не может осуществлять чтение из секции загрузчика; если таблица векторов прерываний расположена в секции прикладной программы, то при выполнении кода из секции загрузчика прерывания запрещены.

BLB02:BLB01 – определяют уровень доступа из секции загрузчика к коду, расположенному в секции прикладной программы.

1 1 – нет никаких ограничений по доступу к коду, расположенному в секции прикладной программы.

1 0 – команда SPM не может осуществлять запись по адресам, находящимся в пределах секции прикладной программы;

0 1 – команда SPM не может осуществлять запись по адресам, находящимся в пределах секции прикладной программы, и команда LPM, вызываемая из секции загрузчика, не может осуществлять чтение из секции прикладной программы; если таблица векторов прерываний расположена в секции загрузчика, то при выполнении кода из секции прикладной программы прерывания запрещены;

0 0 – команда LPM, вызываемая из секции загрузчика, не может осуществлять чтение из секции прикладной программы; если таблица векторов прерываний расположена в секции загрузчика, то при выполнении кода из секции прикладной программы прерывания запрещены.

LB1:LB0 – определяют тип защиты памяти программ и данных.

1 1 – защита кода и данных отключена;

0 1 – запись FLASH и EEPROM запрещена;

0 0 – запрещены запись и чтение FLASH и EEPROM.

Конфигурационные ячейки

Конфигурационные ячейки (Fuse Bits) (рис. 54) определяют параметры конфигурации микроконтроллера. Они расположены в отдельном адресном пространстве, доступном только при программировании. Все конфигурационные ячейки микроконтроллера ATmega8535 сгруппированы в два байта. Для изменения содержимого конфигурационных ячеек используются специальные команды программирования. Команда «Стирание кристалла» на состояние этих ячеек не влияет.

При запрограммированной ячейке защиты LB1 конфигурационные ячейки блокируются. Поэтому конфигурацию микроконтроллера необходимо задавать до программирования ячеек защиты.

S8535C	WDTON	SPIEN	CKOPT	EESAVE	BOOTSZ1	BOOTSZ0	BOOTRST	High
BODLEVEL	BODEN	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0	Low
7	6	5	4	3	2	1	0	Fuse Bits

Рис. 54. Конфигурационные ячейки

S8535C – режим совместимости с микроконтроллером AT90S8535 (старая модификация ATmega8535) (по умолчанию «1» – выключен);

WDTON – сторожевой таймер (по умолчанию «1» – включен);

SPIEN – режим программирования микроконтроллера по интерфейсу SPI (по умолчанию «0» – включен);

CKOPT – дополнительный бит выбора тактового генератора (по умолчанию «1»);

EESAVE – влияние команды «Стирание кристалла» на EEPROM-память (по умолчанию «1» – стирает);

BOOTSZ1:BOOTSZ0 – определяют размер секции загрузчика (табл.13):

Таблица 13

Определение размера секции загрузчика

BOOTSZ1: BOOTSZ0	Размер загрузчика	Стра-ниц	Секция прикладных программ	Секция загрузчика
1 1	128*16	4	0x000 – 0xF7F	0xF80 – 0xFFF
1 0	256*16	8	0x000 – 0xEFF	0xF00 – 0xFFF
0 1	512*16	16	0x000 – 0xDFF	0xE00 – 0xFFF
0 0 (по умолч.)	1024*16	32	0x000 – 0xBFF	0xC00 – 0xFFF

BOOTRST – определяет положение вектора сброса:

1 – адрес 0x0000 (по умолчанию);

0 – адрес начала секции загрузчика;

BODLEVEL – определяет порог срабатывания схемы BOR (по умолчанию – «1»);

BODEN – разрешает/запрещает функционирование схемы BOR:

1 (по умолчанию) – запрещено, микроконтроллер сбрасывается при напряжении питания ниже 2,7В (Uпит = 5В);

0 – разрешено, микроконтроллер сбрасывается при напряжении питания ниже 4,0В (Uпит = 5В);

SUT1:SUT0 – определяет длительность задержки старта:

0 0 – нет дополнительной задержки, включен BOD;

0 1 – дополнительная задержка 4,1 мс при Vcc = 5 В;

1 0 – дополнительная задержка 65 мс при Vcc = 5 В (по умолчанию);

1 1 – зарезервировано;

CKSEL3:CLSEL0 – определяет режим работы тактового генератора:

0000 – внешняя тактовая частота;

0001…0100 – внутренний калиброванный RC-генератор;

0101…1000 – внешний RC-генератор;

1001 – внешний низкочастотный кварцевый генератора;

1010…1111 – внешний высокочастотный кварцевый генератор.

Идентификатор

Все микроконтроллеры фирмы Atmel имеют три 8-битные ячейки, содержимое которых позволяет идентифицировать устройство (Signature Bytes).

1. 0x000: 0x1E (производитель – Atmel)
2. 0x001: 0x93 (8 KB FLASH памяти)
3. 0x002: 0x08 (микроконтроллер ATmega8535 с 8K FLASH памяти)

Ячейки идентификатора расположены в отдельном адресном пространстве, доступ к которому возможен только в режиме программирования. В отличие от конфигурационных ячеек, ячейки идентификатора доступны только для чтения.

Ячейки идентификатора предназначены для связи программирующего устройства с микроконтроллером. Считывание этих ячеек производится при попытке установки связи программатора с микроконтроллером.

Калибровочные ячейки

В калибровочные ячейки при изготовлении микроконтроллера заносятся калибровочные константы, предназначенные для подстройки на номинальную

частоту внутреннего RC-генератора. Микроконтроллер ATmega8535 имеет четыре 8-битных ячейки. Располагаются они в старших байтах адресного пространства ячеек идентификатора по адресам 0x000, 0x001, 0x002 и 0x003 для частот 1, 2, 4 и 8 МГц соответственно.

Генератор автоматически калибруется на частоту 1 МГц занесением соответствующей константы в регистр OSCCAL. При использовании другой частоты RC-генератора его калибровку необходимо осуществлять вручную. Для этого программатор во время программирования должен прочитать содержимое калибровочной ячейки и занести его по какому-либо адресу FLASH-памяти программ. А программа должна после старта считать это значение из памяти программ и занести его в регистр OSCCAL.

Режим параллельного программирования

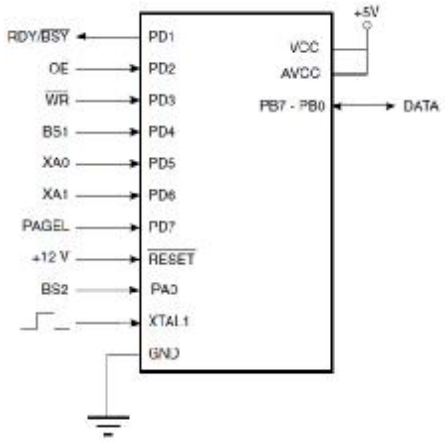


Рис. 55. Параллельное программирование

В режиме параллельного программирования от программатора к микроконтроллеру передаются одновременно все биты кода команды или байта данных. Этот режим задействует большое число выводов микроконтроллера и требует использования дополнительного источника повышенного напряжения (12 В). Программирование в параллельном режиме осуществляется специализированными программаторами. Основное применение этого режима – «прошивка» микроконтроллеров перед установкой их на плату в условиях массового производства.

Схема включения микросхем в режиме параллельного программирования приведена на рис. 55, назначение выводов – в табл. 14.

Команды программирования в параллельном режиме:

1000 0000 – стирание кристалла;

0100 0000 – запись конфигурационных ячеек;

0010 0000 – запись ячеек защиты;

0001 0000 – запись FLASH-памяти;

0001 0001 – запись EEPROM-памяти;

0000 1000 – чтение идентификатора;

0000 0100 – чтение конфигурационных ячеек и ячеек защиты;

00000010 – чтение FLASH-памяти;

00000011 – чтение EEPROM-памяти.

Таблица 14

Назначение выводов микроконтроллера в режиме параллельного программирования

Сигнал	Вывод	Вход/выход	Назначение
RDY/$\overline{\text{BSY}}$	PD1	Выход	Состояние устройства: 0 – занято (выполняется предыдущая команда); 1 – готово к приему следующей команды
$\overline{\text{OE}}$	PD2	Вход	Управление режимом работы шины данных PB7...PB0: 0 – выход, 1 – вход
$\overline{\text{WR}}$	PD3	Вход	Сигнал записи (активный уровень – «0»)
BS1 BS2	PD4 PA0	Вход	Выбор байта 0 – младший байт, 1 – старший байт 0 – младший байт, 1 – дополнительный байт
XA0 XA1	PD5 PD6	Вход	Определяют действие, выполняемое по положительному импульсу на выводе XTAL1 0 0 – загрузка адреса ячейки памяти (байт зависит от сигнала BS1) 0 1 – загрузка данных (байт зависит от BS1) 1 0 – загрузка команды 1 1 – нет действия, режим ожидания
PAGEL	PD7	Вход	Сигнал загрузки страницы памяти
DATA	PB7... PB0	Вход/ выход	Двунаправленная шина данных

Первой операцией при программировании микроконтроллера является его перевод в режим программирования. Для этого необходимо:

1. Подать на микроконтроллер напряжение питания.
2. Подать на вывод RESET напряжение логического «0» и сформировать не менее трех импульсов на выводе XTAL1.
3. Подать на выводы PAGEL, XA1, XA0, BS1 напряжение логического «0» на время не менее 100 нс.
4. Подать напряжение 11,5...12,5 В на вывод RESET и удерживать напряжение логического «0» на выводах PAGEL, XA1, XAO, BS1 в течение, как минимум, 10 мкс. Любая активность на указанных выводах в течение этого времени приведет к тому, что микроконтроллер не перейдет в режим программирования.
5. Выждать не менее 300 мкс перед посылкой следующей команды.

Стирание кристалла

Команда «Стирание кристалла» должна выполняться перед каждым перепрограммированием микроконтроллера. Команда полностью уничтожает содержимое FLASH- и EEPROM-памяти, а затем сбрасывает ячейки защиты (записывает в них «1»). На состояние конфигурационных ячеек данная команда не влияет. Можно предотвратить стирание EEPROM-памяти путем программирования конфигурационной ячейки EESAVE.

Для выполнения команды «Стирание кристалла» необходимо:

1. Загрузить команду «Стирание кристалла» (код 1000 0000).

2. Подать на вывод WR отрицательный импульс; при этом на выводе RDY/BSY появляется сигнал логического «0».
3. Ждать появления на выводе RDY/BSY сигнала логической «1».

Запись FLASH-памяти

FLASH-память микроконтроллера ATmega8535 объемом 4 кбайт организована в виде 128 страниц по 32 адреса. Запись FLASH-памяти производится в следующей последовательности:

1. Загрузить команду «Запись FLASH-памяти» (код 0001 0000).
2. Загрузить младший байт адреса (положение ячейки внутри страницы).
3. Загрузить младший байт данных.
4. Загрузить старший байт данных.
5. Запомнить данные в буфере.
6. Повторить пп. 2...5 до полного заполнения буфера страницы.
7. Загрузить старший байт адреса (номер страницы).
8. Записать страницу.
9. Повторить пп. 2...8 для записи остальных страниц памяти программ.
10. Завершить программирование, загрузив команду «Нет операции» (код 0000 0000).

Чтение FLASH-памяти

Для чтения FLASH-памяти необходимо:

1. Загрузить команду «Чтение FLASH-памяти» (код 0000 0010).
2. Загрузить старший байт адреса.
3. Загрузить младший байт адреса.
4. Сбросить OE и BS1 в «0», после этого с шины данных DATA можно будет считать значение младшего байта содержимого ячейки памяти.
5. Установить BS1 в «1», после этого с шины данных DATA можно будет считать значение старшего байта содержимого ячейки памяти.
6. Установить OE в «1».

Запись EEPROM-памяти

Запись EEPROM-памяти производится в следующей последовательности:

1. Загрузить команду «Запись EEPROM-памяти» (код 0001 0001).
2. Загрузить старший байт адреса.
3. Загрузить младший байт адреса.
4. Загрузить байт данных.
5. Запомнить данные в буфере.
6. Повторить пп. 3...5 до полного заполнения буфера.
7. Записать страницу.

Чтение EEPROM-памяти

Для чтения содержимого EEPROM-памяти необходимо:

1. Загрузить команду «Чтение EEPROM-памяти» (код 0000 0011).
2. Загрузить старший байт адреса.
3. Загрузить младший байт адреса.
4. Сбросить OE и BS1 в «0», после этого с шины данных DATA можно будет считать содержимое ячейки памяти.

5. Установить OE в «1».

Программирование конфигурационных ячеек

Младший конфигурационный байт:

1. Загрузить команду «Запись конфигурационных ячеек» (код 0100 0000).
2. Загрузить младший байт данных. Если бит сброшен в «0», выполняется программирование соответствующей ячейки, если установлен в «1» – ее сброс.
3. Записать младший байт конфигурации.

Старший конфигурационный байт:

1. Загрузить команду «Запись конфигурационных ячеек» (код 0100 0000).
2. Загрузить младший байт данных. Если бит сброшен в «0», выполняется программирование соответствующей ячейки, если установлен в «1» – ее сброс.
3. Записать старший байт конфигурации.

Программирование ячеек защиты

1. Загрузить команду «Запись ячеек защиты» (код 0010 0000).
2. Загрузить младший байт данных. Для программирования ячейки соответствующий бит должен быть сброшен в «0». Неиспользуемые биты должны быть всегда установлены в «1».
3. Записать младший байт конфигурации.

Чтение конфигурационных ячеек и ячеек защиты

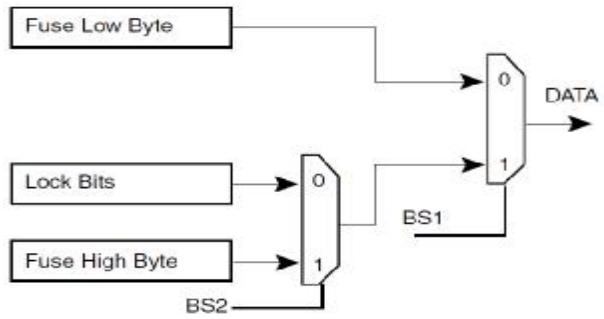


Рис. 56. Чтение конфигурационных ячеек и ячеек защиты

Чтение указанных ячеек выполняется в следующей последовательности (рис. 56):

1. Загрузить команду «Чтение конфигурационных ячеек и ячеек защиты» (код 0000 1000).
2. Сбросить OE, BS1 и BS2 в «0», после этого с шины данных DATA можно будет считать значение младшего конфигурационного байта.
3. Сбросить OE в «0», а BS1 и BS2 установить в «1». После этого с шины данных DATA можно будет считать значение старшего конфигурационного байта.
4. Сбросить OE и BS1 в «0», а BS2 установить в «1». После этого с шины данных DATA можно будет считать значение дополнительного конфигурационного байта.
5. Сбросить OE и BS2 в «0», а BS1 установить в «1». После этого с шины данных DATA можно будет считать значение байта защиты.
6. Установить OE в «1».

Чтение ячеек идентификатора и калибровочных ячеек осуществляется в следующей последовательности:

1. Загрузить команду «Чтение ячеек идентификатора» (код 0000 1000).
2. Загрузить младший байт адреса (0x00...0x02).

3. Сбросить OE и BS1 в «0», после этого с шины данных DATA можно будет считать содержимое выбранной ячейки идентификатора.
4. Установить OE в «1».

Чтение калибровочных констант осуществляется аналогичным образом:

1. Загрузить команду «Чтение ячеек идентификатора» (код 0000 1000).
2. Загрузить младший байт адреса (0x00...0x03).
3. Сбросить OE в «0», а BS1 установить в «1», после этого с шины данных DATA можно будет считать значение выбранной калибровочной константы.
4. Установить OE в «1».

Режим последовательного программирования

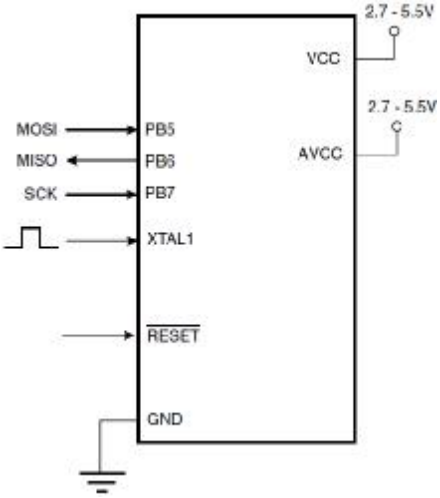


Рис. 57. Схема подключения микроконтроллера при последовательном программировании

В режиме последовательного программирования запись памяти программ и данных осуществляется по последовательному интерфейсу SPI. Этот режим часто используется для программирования (перепрограммирования) микроконтроллера непосредственно в устройстве. Схема включения микросхем в режиме программирования по последовательному каналу приведена на рис. 57.

Для обмена данными между программатором и устройством используются три линии:

SCK (PB7) – тактовый сигнал,

MOSI (PB5) – вход данных,

MISO (PB6) – выход данных.

При программировании по последовательному каналу микроконтроллеру требуется источник тактового сигнала. В качестве тактового может использоваться любой из допустимых для микроконтроллера источников. При этом должно выполняться условие: длительность импульсов как «0», так и «1» уровня сигнала SCK должна быть

- > 2 (при *f*ск < 12 МГц)
- > 3 (при *f*ск > 12 МГц)

периодов тактового сигнала микроконтроллера.

Конт.	Цепь
1	MISO
2	Vcc
3	SCK
4	MOSI
5	$\overline{\text{RESET}}$
6	GND

Рис. 58. Типовая цоколевка разъема программирования

Для внутрисхемного проектирования микроконтроллера по последовательному интерфейсу SPI в принципиальной схеме необходимо предусмотреть разъем программирования, типовая цоколевка которого представлена на рис. 58.

Программирование осуществляется путем посылки 4-байтных команд на вывод MOSI микроконтроллера. Результат выполнения команд чтения снимается с вывода MISO микроконтроллера. Передача команд и выдача результатов их выполнения осуществляются от старшего бита к младшему. При этом «защелкивание» входных данных выполняется по

нарастающему фронту сигнала SCK, а «защелкивание» выходных данных – по спадающему.

Команды режима программирования по последовательному интерфейсу представлены в табл. 15.

Таблица 15

Команды программирования по последовательному интерфейсу

Команда	Формат команды				Описание
	1 байт	2 байт	3 байт	4 байт	
Разрешение программирования	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Разрешает программирование микроконтроллера, пока на выводе $\overline{\text{RESET}}$ присутствует напряжение НИЗКОГО уровня
Стирание кристалла	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Очистка содержимого FLASH и EEPROM-памяти
Чтение памяти программ	0010 H000	0000 aaaa	bbbb bbbb	oooo oooo	Читает H (старший или младший байт) данных o из FLASH памяти по адресу a:b
Загрузка страницы памяти программ	0100 H000	0000 xxxx	xxxb bbbb	iiii iiii	Записывает H (старший или младший байт) данных i в страницу FLASH памяти по адресу b. Младший байт данных записывается первым.
Запись страницы памяти программ	0100 1100	0000 aaaa	bbbx xxxx	xxxx xxxx	Записывает страницу FLASH памяти по адресу a:b
Чтение памяти EEPROM	1010 0000	00xx xxxa	bbbb bbbb	oooo oooo	Считывает данные o из EEPROM-памяти по адресу a:b
Запись памяти EEPROM	1100 0000	00xx xxxa	bbbb bbbb	iiii iiii	Записывает данные i в EEPROM-памяти по адресу a:b
Чтение Lock Bits	0101 1000	0000 0000	xxxx xxxx	xxoo oooo	Считывает Lock bits
Запись Lock Bits	1010 1100	111x xxxx	xxxx xxxx	11ii iiii	Записывает Lock bits
Чтение байта идентификатора	0011 0000	00xx xxxx	xxxx xxbb	oooo oooo	Считывает байт идентификатора o по адресу b
Запись младшего байта Fuse Bits	1010 1100	1010 0000	xxxx xxxx	iiii iiii	Записывает Fuse Bits 0 – программирование бита, 1 – стирание бита
Запись старшего байта Fuse Bits	1010 1100	1010 1000	xxxx xxxx	iiii iiii	
Чтение младшего байта Fuse Bits	0101 0000	0000 0000	xxxx xxxx	oooo ooo	Считывает Fuse Bits 0 – запрограммирован, 1 – не запрограммирован
Чтение старшего байта Fuse Bits	0101 1000	0000 1000	xxxx xxxx	oooo ooo	
Чтение калибровочного байта	0011 1000	00xx xxxx	0000 00bb	oooo ooo	o – калибровочный байт

Для перевода микроконтроллера в режим программирования по последовательному каналу необходимо:

1. Подать на микроконтроллер напряжение питания, при этом на выводах SCK и $\overline{\text{RESET}}$ должно присутствовать напряжение логического «0». В

некоторых случаях (если программатор не гарантирует установку сигнала SCK в «0» при подаче питания) после сброса сигнала SCK в «0» необходимо подать на вывод $\overline{\text{RESET}}$ положительный импульс длительностью не менее 2 периодов тактового сигнала микроконтроллера.

2. Выждать не менее 20 мс.
3. Послать на вывод MOSI команду «Разрешение программирования».

Для контроля прохождения команды при посылке 3-го байта возвращается значение 2-го байта (0x53). Если возвращаемое значение отлично от указанного, необходимо подать на вывод $\overline{\text{RESET}}$ положительный импульс и снова послать команду «Разрешение программирования». Причем независимо от возвращаемого значения необходимо передавать все 4 байта команды. Отсутствие возврата числа 0x53 после нескольких попыток указывает на отсутствие связи между программатором и микросхемой либо на неисправность микросхемы. После завершения программирования на вывод $\overline{\text{RESET}}$ можно подать напряжение уровня логической «1» для перевода микроконтроллера в рабочий режим либо выключить его. В последнем случае необходимо:

1. Подать на вывод XTAL1 напряжение логического «0», если тактирование микроконтроллера осуществляется от внешней схемы.
2. Подать на вывод $\overline{\text{RESET}}$ напряжение высокого уровня.
3. Отключить напряжение питания от микроконтроллера.

Программирование FLASH-памяти программ осуществляется постранично. Сначала содержимое страницы побайтно заносится в буфер по командам «Загрузка страницы FLASH-памяти». В каждой команде передаются младшие биты адреса изменяемой ячейки (положение ячейки внутри страницы) и записываемое значение. Содержимое каждой ячейки должно загружаться в следующей последовательности: сначала младший байт, потом старший. Фактическое программирование страницы FLASH-памяти осуществляется после загрузки буфера страницы по команде «Запись страницы FLASH-памяти». В команде передаются старшие биты адреса ячеек (номер страницы). Дальнейшее программирование памяти можно будет выполнять только после завершения записи страницы. Определить момент окончания записи можно:

- выдерживать между посылкой команд паузу длительностью не менее 4,5 мс;
- контролировать содержимое любой из записываемых ячеек после посылки команды записи: до завершения записи ячейки при ее чтении возвращается значение 0xFF, а после завершения – записанное значение.

Программирование EEPROM памяти

Программирование EEPROM-памяти осуществляется побайтно.

Режим самопрограммирования

Все микроконтроллеры семейства ATmega имеют возможность самопрограммирования, т. е. самостоятельного изменения содержимого своей памяти программ. Эта особенность позволяет создавать на их основе очень гибкие системы, алгоритм работы которых будет меняться самим микроконтроллером в зависимости от каких-либо внутренних условий или внешних событий.

Вся область памяти программ микроконтроллера ATmega8535 логически разделена на две секции – секцию прикладной программы (Application Section) и секцию загрузчика (Boot Loader Section). Изменение памяти программ осуществляется программой-загрузчиком, расположенной в одноименной секции. Для загрузки нового содержимого памяти программ, а также для выгрузки старого, программа-загрузчик может использовать любой интерфейс передачи данных (USART, SPI, TWI), имеющийся в составе микроконтроллера. Загрузчик может изменять содержимое обеих секций. Это позволяет ему модифицировать собственный код и даже удалять себя из памяти, если надобность в нем отпадет. Уровень доступа (чтение, запись) к каждой из секций задается пользователем с помощью ячеек защиты BLB02:BLB01 и BLB12:BLB11.

Переход к программе-загрузчику может осуществляться следующими способами:

- вызов из основной программы командами CALL/JMP;
- получение команды перехода к программе-загрузчику по интерфейсу USART или SPI;
- перемещение вектора сброса в начало секции загрузчика, в этом случае запуск программы-загрузчика будет осуществляться автоматически после каждого сброса микроконтроллера.

Размер секции загрузчика и соответственно размер секции прикладной программы задается с помощью двух конфигурационных ячеек – BOOTSZ1:BOOTSZ0.

Помимо разбиения памяти программ на секции, имеется также разделение памяти программ на две области фиксированного размера, называемые «чтение при записи» (Read-While-Write – RWW (0x000 – 0xBFF)) и «нет чтения при записи» (No Read-While-Write – NRWW(0xC00 – 0xFFF)).

Отличие между этими областями заключается в различном поведении центрального процессора при изменении расположенных в них данных:

- во время выполнения операции стирания или записи страницы памяти программ, расположенной в области RWW, процессор может осуществлять чтение из области NRWW;
- во время выполнения операции стирания или записи страницы памяти программ, расположенной в области NRWW, процессор останавливается до окончания этой операции.

Таким образом, во время изменения содержимого страницы памяти программ, расположенной в области RWW, чтение этой области запрещено. Попытка обратиться во время программирования к коду, находящемуся в области RWW (в

результате выполнения команд CALL/JMP/LPM или в результате прерывания), может привести к непредсказуемым последствиям. Во избежание этого следует либо запретить прерывания, либо перенести таблицу векторов прерываний в секцию загрузчика, которая всегда находится в области NRWW.

Для управления режимом самопрограммирования используется регистр SPMCR (рис. 59).

SPMIE	RWWSB	-	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	SPMCR
7	6	5	4	3	2	1	0	

Рис. 59. Регистр управления режимом самопрограммирования SPMCR

SPMIE – разрешение прерывания «Готовность SPM». Если в этом бите записана «1», и флаг I регистра SREG установлен в «1», то разрешается прерывание «Готовность SPM», которое генерируется все время, пока бит SPMEN регистра сброшен в «0».

RWWSB – запрет доступа к области RWW, показывает возможность обращения по адресам, расположенным в области RWW: «1» – доступ к области RWW запрещен, «0» – разрешен; установка флага осуществляется аппаратно при выполнении операций записи или стирания страницы памяти, сброс флага осуществляется либо программно, записью «1» в бит RWWSRE по окончании операции, либо аппаратно, при запуске операции загрузки страницы.

RWWSRE – чтение области RWW разрешено, одновременная установка этого бита и бита SPMEN позволяет разрешить доступ к области RWW, разрешение доступа к этой области осуществляется запуском команды SPM в течение 4 тактов после установки указанных битов, разрешение доступа к области RWW может осуществляться только после завершения операции программирования (после сброса флага SPMEN).

BLBSET – изменение ячеек защиты загрузчика, при одновременной установке этого бита и бита SPMEN команда SPM, запущенная в течение последующих 4 тактов, осуществит установку защитных ячеек загрузчика в соответствии с содержимым регистра R0; сброс бита BLBSET осуществляется аппаратно после установки ячеек защиты либо по истечении указанного времени. По команде LPM, запущенной в течение 3 тактов после установки указанных битов, будет осуществлено чтение либо конфигурационных ячеек, либо ячеек защиты (зависит от значения бита Z0 регистра Z).

PGWRT – запись страницы FLASH-памяти, при одновременной установке этого бита и бита SPMEN команда SPM, запущенная в течение последующих 4 тактов, осуществит запись страницы памяти программ из временного буфера. Адрес страницы должен быть загружен в старший байт регистра Z (R31). Сброс бита PGWRT осуществляется аппаратно по окончании записи страницы либо по истечении указанного времени. При записи в секцию NRWW центральный процессор останавливается на время выполнения операции.

PGERS – стирание страницы, при одновременной установке этого бита и бита SPMEN команда SPM, запущенная в течение последующих 4 тактов, осуществит стирание страницы памяти программ. Адрес страницы должен быть загружен в старший байт регистра Z (R31). Сброс бита PGERS осуществляется аппаратно по окончании стирания страницы либо по истечении указанного времени. При записи в секцию NRWW центральный процессор останавливается на время выполнения операции.

SPMEN – разрешение выполнения команды SPM. Установка этого бита разрешает запуск команды SPM в течение последующих 4 тактов. Если бит SPMEN устанавливается одновременно с одним из битов RWWSRE, BLBSET, PGWRT или PGERS, выполняется операция, определяемая этим битом, если устанавливается только бит SPMEN, осуществляется сохранение содержимого регистров Rl:R0 во временном буфере по адресу, находящемуся в регистре Z (младший байт регистра игнорируется). Сброс бита SPMEN осуществляется аппаратно после завершения операции либо по истечении указанного времени

Во время записи в EEPROM-память изменение регистра SPMCR невозможно. Поэтому перед тем, как записать какое-либо значение в регистр SPMCR, рекомендуется дождаться сброса флага EEWE регистра EECR.

Для адресации памяти программ при использовании команды SPM используется индексный регистр Z, получаемый объединением двух регистров общего назначения – R30 (младший байт) и R31 (старший байт) (рис. 60).

Z15	Z14	Z13	Z12	Z11	Z10	Z9	Z8	ZH (R31)
Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	ZL (R30)
7	6	5	4	3	2	1	0	

Рис. 60. Индексный регистр Z

Поскольку память программ в микроконтроллере имеет страничную организацию, счетчик команд можно условно разбить на две части. Первая часть (младшие биты) адресует ячейку на странице, а вторая часть определяет страницу. После запуска операции программирования содержимое регистра Z фиксируется, и его можно использовать для других целей.

Изменение содержимого памяти программ осуществляется в следующей последовательности:

1. Заполнение временного буфера страницы новым содержимым.
2. Стирание страницы.
3. Перенос содержимого буфера в память программ.

Стирание страницы может выполняться как после заполнения буфера, так и перед его заполнением. Если необходимо изменить только часть страницы, приведенный порядок действий является единственно возможным. В этом случае содержимое ячеек, не требующих изменения, сохраняется в буфере перед очисткой страницы.

Для определения момента окончания выполнения операций можно либо опрашивать состояние флага SPMEN регистра SPMCR, дожидаясь его сброса, либо воспользоваться прерыванием «Готовность SPM». Это прерывание генерируется все время, пока флаг SPMEN сброшен. В последнем случае таблица векторов прерываний должна находиться в секции загрузчика, а это прерывание должно быть разрешено установкой флага SPMIE регистра SPMCR. Операция записи FLASH-памяти тактируется внутренним RC-генератором. Время записи страницы FLASH-памяти составляет 3,7…4,5 мс.

Для стирания страницы памяти программ необходимо занести адрес страницы в регистр Z (секция PCPAGE), записать значение 0000011 в регистр SPMCR и в течение последующих четырех тактов выполнить команду SPM. Содержимое регистров R1 и R0 при этом игнорируется.

Для занесения слова команды в буфер следует загрузить адрес ячейки в регистр Z (секция PCWORD), а код операции – в регистры R1:R0. После этого необходимо записать значение 0000011 в регистр SPMCR и в течение последующих четырех тактов выполнить команду SPM. Очистка буфера осуществляется автоматически, по окончании записи страницы, либо вручную, записью «1» в бит RWWSRE регистра SPMCR. Запись по одному и тому же адресу в буфере невозможна без его очистки.

Запись содержимого буфера в память программ осуществляется аналогично. В регистр Z (секция PCPAGE) заносится адрес страницы, в регистр SPMCR записывается значение 0000101, и в течение последующих четырех тактов выполняется команда SPM. Содержимое регистров R1 и R0 при этом игнорируется.

Изменение ячеек защиты загрузчика BLB12:BLB11 и BLB02:BLB01 осуществляется командой SPM. Для этого необходимо загрузить в регистр R0 требуемое значение регистра Lock Bits (сброшенный бит означает программирование соответствующей ячейки). После этого необходимо записать значение 0001001 в регистр SPMCR и в течение последующих четырех тактов выполнить команду SPM. Содержимое регистра Z при этом игнорируется, однако для совместимости с будущими устройствами рекомендуется записывать в него значение 0x0001. Во время программирования ячеек защиты можно обращаться к любой области памяти программ.

Загрузчик может также считывать содержимое конфигурационных ячеек и ячеек защиты. Для чтения байта защиты следует загрузить в регистр Z число 0x0001, записать в регистр SPMCR значение 0001001 и в течение трех последующих тактов выполнить команду LPM. В результате содержимое байта защиты будет занесено в заданный регистр общего назначения. Чтение конфигурационных байтов осуществляется аналогично. В регистр Z загружается адрес байта (0x0000 – младший байт, 0x0003 – старший байт, 0x0002 – дополнительный байт), после чего в регистр SPMCR следует записать значение 0001001 и в течение трех последующих тактов выполнить команду LPM. В результате выполнения команды содержимое выбранного байта конфигурации будет занесено в регистр общего назначения.

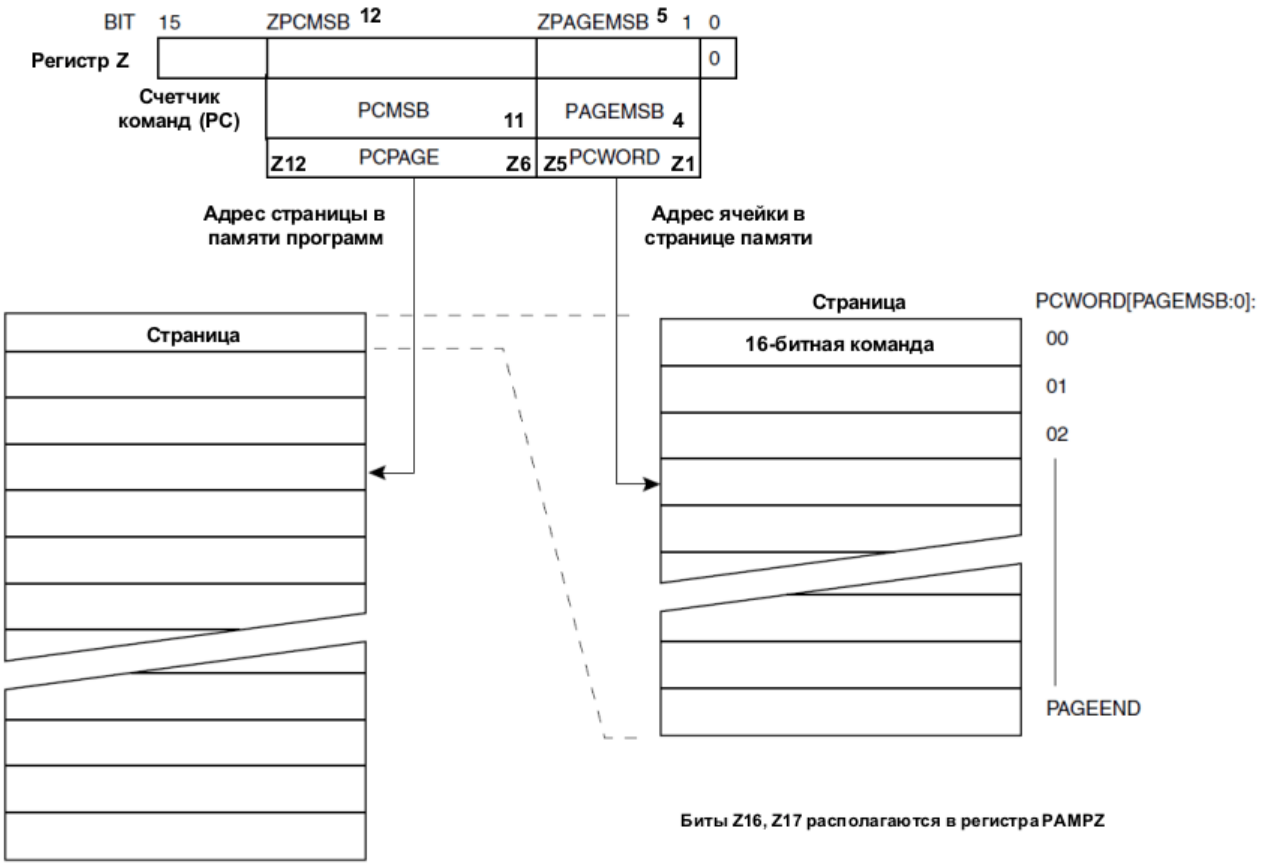


Рис. 61. Адресация памяти программ при использовании команды SPM

Для предотвращения повреждения кода в режиме самопрограммирования при внезапном пропадании питания рекомендуется:

1. При отсутствии необходимости модификации программы загрузчика запрещать его изменение битами защиты.
2. При понижении напряжения питания удерживать сигнал сброса процессора RESET равным логическому «0»: это достигается включением схемы аппаратного монитора питания (схемы BOD).
3. Во время действия низкого напряжения переводить микроконтроллер в режим пониженного энергопотребления (SLEEP).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Программирование микроконтроллеров ATmega8535: методические указания к выполнению лабораторных работ / Р.З. Хусаинов, В.Б. Садов, Д.Н. Тагиров, А.А. Бунаков. – Челябинск, Изд-во ЮУрГУ, 2007.
2. Евстифеев, А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL / А.В. Евстифеев. – М.: издательский дом «ДОДЭКА-XXI», 2004.
3. Евстифеев, А.В. Микроконтроллеры AVR семейства Mega: руководство пользователя/ А.В. Евстифеев. – М.: издательский дом «ДОДЭКА-XXI», 2007.
4. Трамперт, В. AVR-RISC микроконтроллеры: архитектура, аппаратные ресурсы, система команд, программирование, применение / В. Трамперт; пер. с нем. В.П. Репало и др. – Киев: МК-Пресс, 2006.
5. Мортон, Д. Микроконтроллеры AVR: ввод. курс: пер. с англ. / Д. Мортон. – М.: ДОДЭКА-21, 2006
6. www.atmel.com