

## Глава 3

# Компьютер и микроконтроллеры

---

Широкое применение радиоэлектронной автоматики во всех сферах нашей жизни вызывает необходимость применения специальных устройств — микроконтроллеров. Практически каждый из бытовых приборов и машин, имеющих автоматическое управление, например, телефон, телевизионный приемник, стиральная машина и другие содержат в своем составе микроконтроллер.

Материал этой главы разбит как бы на три взаимосвязанных составных части.

Сначала кратко познакомимся с устройством микроконтроллеров, затем будет рассмотрен процесс создания и отладки программы для работы микроконтроллера, а в заключении будет рассмотрен процесс ввода программы в память микроконтроллера, и будут приведены описания компьютерных программных средств, предназначенных для программирования микроконтроллеров.

---

Каждый микроконтроллер, образно говоря, является миниатюрным подобием компьютера и имеет в своем составе и элементы памяти, и счетно-решающее устройство, и шину передачи команд и шину передачи данных. Для нормальной работы микроконтроллера, так же как и для компьютера, требуется специальная программа. Созданию таких программ, предназначенных для управления микроконтроллером, преобразованию кодов этих программ в удобный для контроллера вид, а также размещению программ в памяти контроллера посвящен материал этой главы.

Каждая из компьютерных программ (в том числе и программы для микроконтроллеров) в процессе своего создания проходит несколько этапов.

---

- На первом этапе с помощью соответствующего языка программирования пишутся (создаются) исходные коды программы. При этом обычно используется либо язык Ассемблера, либо язык СИ. Ассемблер требует детального указания для выполнения самых простейших, элементарных, действий. Каждой команде Ассемблера соответствует одно элементарное действие, поэтому исходные коды на этом языке программирования получаются очень громоздкими. К тому же программисту нужно очень хорошо знать устройство микроконтроллера и взаимодействие всех его составляющих частей. Ассемблер — «язык программирования низкого уровня». Файл с исходными кодами программы на языке Ассемблера всегда имеет расширение `ASM`. Команды языка программирования СИ зачастую обозначают выполнение сразу нескольких элементарных действий, при этом программисту не всегда нужно знать тонкости взаимодействия составных частей микроконтроллера. Такой язык программирования называется языком «высокого уровня». Файл с исходными кодами программы на языке СИ всегда имеет расширение `C` или `CPP`. Создаются исходные коды программы в специальном текстовом редакторе.
- После написания исходных кодов и проверки их на предмет отсутствия ошибок, программу следует преобразовать в тот вид, который может дать необходимый

результат. Если делаем программу для компьютера, то должны исходные коды преобразовать в исполняемый файл. Такие файлы обычно имеют расширение `.exe`. Если делается программа для микроконтроллера, то исходные коды должны быть преобразованы в файл с расширением `.hex`. Для подобных преобразований исходных кодов служит программа, называемая «компилятор». В настоящее время разработаны сложные программные продукты, называемые «среда программирования», которые включают в свой состав и текстовый редактор, и компилятор и отладчик.

- Третий этап работы относится к программе для микроконтроллера и заключается в том, чтобы разместить созданный программный файл в памяти микроконтроллера. Такой процесс происходит с применением специального аппаратного устройства, называемого *программатором*, которое устанавливается между компьютером и микроконтроллером и служит для согласования уровней сигналов этих двух аппаратов. Процесс размещения файла данных в памяти микроконтроллера с использованием программатора носит название «программирование микроконтроллера».

## Основы процесса программирования

Чтобы понять основы процесса размещения необходимых данных из файла в памяти микроконтроллера, рассмотрим подробнее сам процесс программирования.

---

Процесс программирования микроконтроллера в основном ничем не отличается от процесса программирования микросхем ПЗУ (Постоянное Запоминающее Устройство), издавна применяемых радиолюбителями в своих конструкциях, поэтому, по моему мнению, для лучшего понимания процесса полезнее будет вспомнить о программировании ПЗУ.

---

Наибольшей популярностью у радиолюбителей пользовались и пользуются **программируемые** ПЗУ (ППЗУ) с плавкими перемычками и **репрограммируемые** ПЗУ (РПЗУ) со стиранием информации ультрафиолетовым излучением. Первые более доступны, но допускают однократное программирование и имеют меньшую информационную плотность. Вторые значительно дороже, но это окупается их большей информационной емкостью и, что особенно привлекательно для радиолюбителя, возможностью многократного программирования.

Однако на каком бы физическом принципе не была основана работа ПЗУ, для занесения в него данных необходимо специальное устройство — программатор. Промышленность выпускает такие устройства, но они недоступны большинству радиолюбителей. В то же время имеется целый ряд любительских разработок, аппаратных устройств и компьютерных программ, которые позволяют в домашних условиях, с помощью вашего ПК выполнить программирование микроконтроллера (и микросхемы ПЗУ). Большой вклад в создание подобных разработок внесли фирмы-производители микроконтроллеров, заинтересованные в улучшении сбыта этих устройств.

Но прежде чем описывать устройство программатора, познакомимся с устройством большой интегральной схемы (БИС) ПЗУ и теми процессами, которые происходят внутри ПЗУ при программировании. В качестве примера рассмотрим такое

распространенное ПЗУ как K573РФ1, схематическое изображение которого показано на рис. 3.1.

### Внутреннее строение ПЗУ

Структурные схемы всех ПЗУ в первом приближении можно считать одинаковыми, поэтому их устройство рассмотрим на примере популярной среди радиолюбителей микросхемы K573РФ1.

Основу ПЗУ составляет непосредственно накопитель информации, занимающий почти всю площадь кристалла. Вдоль и поперек кристалла проложены непересекающиеся между собой шины, образующие прямоугольную матрицу, в узлы которой включены запоминающие элементы в виде МОП-транзисторов специальной структуры. Схематически внутреннее строение микросхемы ПЗУ показано на рис. 3.2.

Логические сигналы на одну систему шин (например, горизонтальные шины) поступают с выхода дешифратора части адресных разрядов. При этом высокий уровень устанавливается только на одной из этих шин, и, следовательно, если какие-либо транзисторы в выбранном ряду открыты, то такой же уровень установится и на соответствующих вертикальных шинах.

Сигналы вертикальных шин поступают на входы мультиплексора, подключающего к выходу микросхемы только одну из шин. Какая из шин будет подключена к выходу, зависит от значения разрядов адреса, неиспользуемых для дешифрования горизонтальных шин. Следовательно, при каждой комбинации сигналов на адресных входах БИС выходной сигнал определяется состоянием одного из транзисторов, находящегося на перекрестии шин, выбранных кодами X и Y.

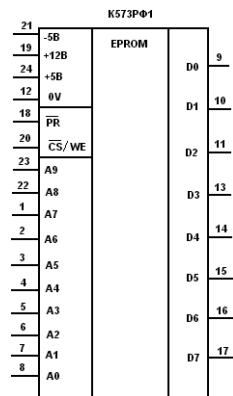


Рис. 3.1. Схема выводов ПЗУ K573РФ1

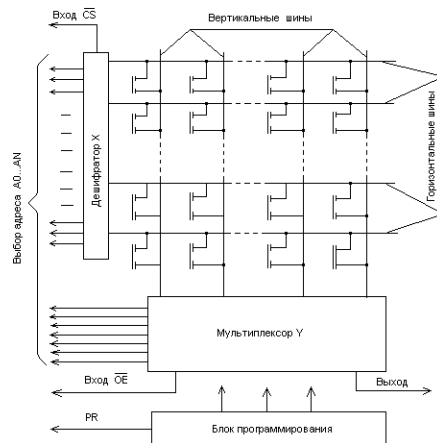


Рис. 3.2. Схема строения ПЗУ

Для простоты изложения материала мы не рассматривали буферные усилители на входах и выходах микросхемы, а также ряд других узлов БИС, несущественных для понимания принципа их работы.

Запоминающая способность РПЗУ с электрическим программированием и стиранием информации ультрафиолетовым излучением определяется особенностями транзисторных структур, входящих в матрицу запоминающих элементов. В отличие от обычных МОП-транзисторов они имеют затвор из поликристаллического кремния, расположенный в слое его окисла, благодаря чему такой затвор и получил название «плавающего»<sup>1</sup>.

<sup>1</sup> Запоминающий МОП-транзистор БИС K573РФ1, кроме плавающего, имеет еще один затвор, предназначенный только для считывания данных.

Запись информации (заряд затвора) происходит в результате приложения между стоком и истоком большего (около 30 В) отрицательного напряжения. Электроны, получающие в электрическом поле достаточную энергию, преодолевают потенциальный барьер на границе полупроводника и окисла и дрейфуют к плавающему затвору, заряжая его отрицательно. Этот заряд создает в канале электрическое поле, стремящееся открыть МОП-транзистор.

После отключения программирующего напряжения заряд затвора не исчезает, поскольку двуокись кремния настолько хороший изолятор, что даже по истечении 10 лет на нем сохраняется более 70% заряда, а электрических соединений, через которые затвор мог бы разрядиться, нет.

Однако если МОП-транзистор с плавающим затвором подвергнуть воздействию ультрафиолетовых лучей, электроны получают энергию, достаточную для того, чтобы «перескочить» обратно на подложку. Затвор при этом разрядится, и запоминающий элемент возвратится в свое первоначальное незапрограммированное состояние. Остается лишь добавить, что подачей программирующих напряжений на транзисторы матрицы управляет специальный блок, также расположенный на кристалле РПЗУ.

Структурная схема БИС ПЗУ с плавкими перемычками, как две капли воды, похожа на уже рассмотренную нами — принципиальная разница заключается в построении запоминающего элемента. Как и в БИС РПЗУ, в узлы матрицы накопителя входят «транзисторы», но самые обыкновенные, а в их эмиттерные (или коллекторные) цепи включены тонкие (20–30 нм) нихромовые перемычки, благодаря которым БИС и приобретает способность «запоминания» информации. Схематическое изображение транзистора такой матрицы показано на рис. 3.3.

При подаче программирующего напряжения перемычка расплавляется, что соответствует записанному биту информации.

К сожалению, ПЗУ с нихромовыми перемычками обладают серьезным недостатком: по прошествии определенного времени некоторые из разрушенных соединений вновь становятся проводящими. Однако этого недостатка удастся избежать, если вместо нихрома в качестве материала плавкой перемычки использовать поликристаллический кремний.

Процесс программирования РПЗУ достаточно прост и состоит из следующих этапов:

- ♦ на адресных входах микросхемы следует установить адрес ячейки, в которую необходимо занести информацию;
- ♦ на выходах данных микросхемы (в процессе программирования выходы БИС ПЗУ являются входами!) следует установить те данные, которые необходимо записать в ячейку;
- ♦ на специальный вход PR микросхемы необходимо подавать импульс управления программированием.

Например, для программирования микросхемы К573РФ1 необходимо повысить напряжение на входе CS/WE (вывод 20) до 12 В, активизируя этим блок программирования. Затем, устанавливая последовательные значения кода адреса на выво-

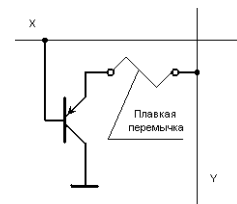


Рис. 3.3. Схема транзисторного узла с плавкой перемычкой

дах A0-A9 и необходимые данные на выводах D0-D7, подать импульсы программирования амплитудой 25 В на вход программирования PR (вывод 18). Импульсы можно подавать только тогда, когда уровни сигналов на входах A0-A9 и D0-D7 стабильны. Для надежной записи информации в каждую ячейку необходимо подать не менее 500–100 импульсов программирования длительностью 1 мс.

Остальные микросхемы этой серии программируют импульсом с уровнем ТТЛ, подаваемым на вход PR. В зависимости от типа БИС для зарядки плавающих затворов одной ячейки достаточно 20–50 миллисекунд. Чтобы программирование произошло, на входе питания блока программирования ( $U_{PR}$ ) должно присутствовать напряжение 18–25 В.

---

После окончания импульса программирования необходимо считать данные и убедиться в правильности записи. Современные БИС РПЗУ допускают считывание информации без отключения источника питания  $U_{PR}$ .

---

Методика программирования ПЗУ с плавкими перемычками принципиально ничем не отличается от алгоритма записи информации в БИС РПЗУ. Разница состоит лишь в том, что требуется более мощный источник программирующего напряжения и за один цикл может быть запрограммирован только один разряд ПЗУ.

## Что такое микроконтроллер

В 1976 г. бурное развитие полупроводниковой технологии привело к созданию фирмой Intel первого микроконтроллера МК-8048. Помимо *центрального процессора* (ЦП), в его состав входила *память программ, память данных*, восьмибитный *таймер* и *27 линий ввода/вывода*. В настоящее время МК-8048 является уже достоянием истории, а вот выпущенные фирмой Intel в 1980 г. микроконтроллеры МК-8051 можно встретить и до сих пор.

Этот микроконтроллер можно считать классическим образцом, по образу и подобию которого позднее было создано множество других изделий. Структурная схема одного из первых представителей семейства микроконтроллеров представлена на рис. 3.4.

Центральный процессор (ЦП) — главный узел микроконтроллера. С ним связано такое важнейшее понятие, как система команд.

**Система команд** — это уникальный, характерный для данного ЦП набор двоичных кодов, определяющих перечень всех его возможных операций. Каждый такой код определяет одну операцию и называется **кодом операции** или командой. Чем больше кодов используется в системе команд, тем больше операций способен выполнить ЦП. МК-8051 является восьмиразрядным, поэтому коды операций у него имеют размер 8 бит. Теоретически может быть всего 256 восьмибитных кодов операций. В МК-8051 используются 255 различных кодов.

В зависимости от числа использованных кодов операций системы команд подразделяют на две группы: CISC и RISC. Термин CISC означает сложную систему команд и является аббревиатурой английского определения Complex Instruction Set Computer. Аналогично термин RISC означает сокращенную систему команд и про-

исходит от английского Reduced Instruction Set Computer. Систему команд МК-8051 можно отнести к типу CISC.

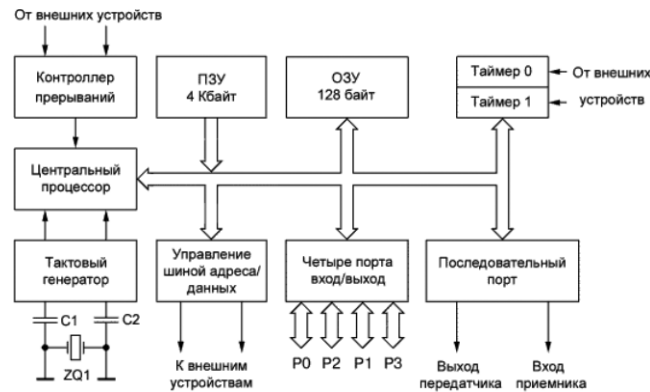


Рис. 3.4. Структурная схема одного из первых микроконтроллеров

Основная идея RISC-архитектуры — это тщательный подбор таких комбинаций кодов операций, которые можно было бы выполнить за один такт тактового генератора. Основной выигрыш от такого подхода — резкое упрощение аппаратной реализации ЦП и возможность значительно повысить его производительность.

Первоначально реализовывать такой подход удавалось, лишь существенно сократив набор команд, отсюда и родилось название RISC. Например, система команд микроконтроллер семейства Microchip PIC16 включает в себя 35 инструкций и может быть отнесена к типу RISC. Очевидно, что в общем случае одной команде CISC-архитектуры должны соответствовать несколько команд RISC-архитектуры. Однако обычно выигрыш от повышения быстродействия в рамках RISC-архитектуры перекрывает потери от менее эффективной системы команд, что приводит к более высокой эффективности RISC-систем в целом по сравнению с CISC. Так, самая быстрая команда МК-8051 выполняется за 12 тактов. Даже если для каждой инструкции потребуется выполнить три инструкции RISC-контроллера, то в итоге RISC-архитектура обеспечит четырехкратное увеличение производительности.

Попутно RISC-архитектура позволяет решить еще ряд задач. Ведь с упрощением ЦП уменьшается число транзисторов, необходимых для его реализации, следовательно, уменьшается площадь кристалла. А с этим связано снижение стоимости и потребляемой мощности.

Например, микроконтроллеры семейства AVR фирмы Atmel имеют систему команд из 120 инструкций, что соответствует типу CISC. Однако большинство из них выполняется за один такт, что является признаком RISC-архитектуры. Сегодня принято считать, что основным признаком RISC-архитектуры является выполнение команд за один такт тактового генератора. Число команд само по себе значения уже не имеет.

**Тактовый генератор** вырабатывает импульсы для синхронизации работы всех узлов устройства. Частоту их следования могут задавать кварцевый резонатор или RC-цепь, подключаемые к выводам микроконтроллера. В некоторых микрокон-

троллерах предусмотрен режим работы тактового генератора без применения внешних элементов. В этом случае частота тактовых импульсов зависит от параметров кристалла, определяемых в процессе его производства.

**ПЗУ** — постоянное запоминающее устройство, предназначенное для хранения программ, поэтому часто эту память называют кодовой или памятью программ. До недавнего времени существовало две основных разновидности ПЗУ — **масочные** и **программируемые**.

---

В масочные ПЗУ информацию заносят в процессе изготовления микроконтроллеров с помощью технологических шаблонов — масок. Изменить ее после окончания производственного цикла невозможно. Такие ПЗУ используют лишь в случаях, когда качество программы не вызывает сомнений и существует массовая потребность в микроконтроллерах именно с этой программой. Достоинство масочных ПЗУ — самая низкая стоимость при массовом производстве (от нескольких тыс. штук).

---

В программируемые ПЗУ информацию записывают с помощью устройства, называемого *программатором*. Это позволяет разработчику доводить программное обеспечение до совершенства, прежде чем он решится на заказ большой партии масочных микросхем. Микроконтроллеры с программируемым ПЗУ выпускаются в двух вариантах корпусов: с «окном» и без него. Корпус с «окном» дает возможность стирать содержимое памяти под воздействием ультрафиолетового излучения и записывать новую информацию с помощью программатора, что обеспечивает многократное использование микросхемы в процессе отладки программного обеспечения. Недостаток такого исполнения программной памяти — очень высокая стоимость микроконтроллеров.

В настоящее время все более популярной становится новая технология реализации ПЗУ — **FLASH-память**. Ее главное достоинство в том, что она построена на принципе электрической перепрограммируемости, т.е. допускает многократное стирание и запись информации с помощью программаторов. Минимальное гарантированное число циклов записи/стирания обычно превышает несколько тысяч. Это существенно увеличивает жизненный цикл и повышает гибкость микроконтроллерных систем, так как позволяет вносить изменения в программу микроконтроллера как на этапе разработки системы, так и в процессе его работы в реальном устройстве.

**ОЗУ** — оперативное запоминающее устройство, используемое для хранения данных, поэтому эту память называют еще памятью данных. Число циклов чтения и записи в ОЗУ не ограничено, но при отключении питающего напряжения вся информация теряется.

Архитектура МК-8051 предполагает раздельное использование памяти программ и памяти данных и носит название **гарвардской**. Обычно такую архитектуру используют для повышения быстродействия системы за счет разделения путей доступа к памяти программ и данных, но в МК-8051 она была применена с целью получения памяти программ и данных, не требующих одинакового размера

**Таймеры T0, T1** — шестнадцатиразрядные программируемые таймеры/счетчики, которые могут быть запрограммированы на выполнение целого ряда функций:

- ♦ их можно использовать для точного формирования временных интервалов;
- ♦ подсчета импульсов на выводах микроконтроллеров;
- ♦ формирования последовательности импульсов;
- ♦ тактирования приемопередатчика последовательного канала связи.

Таймеры/счетчики способны вырабатывать запросы прерываний, переключая ЦП на их обслуживание по событиям и освобождая его от необходимости периодического опроса состояния таймеров. Поскольку основное применение микроконтроллеров находят в системах реального времени, таймеры/счетчики являются их обязательными элементами. В некоторых модификациях число таймеров достигает 32.

**Последовательный порт** — канал информационного обмена микроконтроллеров с внешним миром. Такие каналы связи занимают наименьшее число выводов кристалла, обеспечивая связь на значительные расстояния с минимальными аппаратными затратами. В МК-8051 реализован универсальный асинхронный последовательный приемопередатчик (UART), поддерживающий протокол стандарта RS-232C, что обеспечивает возможность организации связи этого микроконтроллера с персональным компьютером. Кроме RS-232C, популярными протоколами в мире встраиваемых систем являются RS-485, I2C (двухпроводная двунаправленная шина), SPI (последовательный периферийный трехпроводный интерфейс), Bitbus (последовательная магистраль управления), CAN (межконтроллерный сетевой интерфейс), USB (универсальная последовательная шина) и некоторые другие. Практически для любого типа последовательного канала сегодня можно найти микроконтроллер, имеющий в своем составе соответствующий последовательный порт.

**Параллельные порты** ввода/вывода — также обязательная часть любого микроконтроллера. Обычно их используют для связи с ближайшим окружением — датчиками и исполнительными механизмами. Число портов ввода/вывода напрямую зависит от числа выводов корпуса. Существуют микроконтроллеры, у которых их более 100.

Важная особенность параллельных портов микроконтроллера — возможность программирования на выполнение нескольких функций. Например, в МК-8051 выходы портов P0 и P2 могут использоваться либо как обычные статические регистры ввода/вывода, либо в качестве шины адреса и данных для подключения **внешних устройств**, таких как дополнительная память программ, память данных, устройства ввода/вывода. Это придает микроконтроллеру архитектурную гибкость. Порт P3 может использоваться как статический регистр ввода/вывода, либо выполнять специальные функции для работы последовательного канала, таймеров, контроллера прерываний и т. д. Возможность перепрограммирования позволяет с максимальной эффективностью задействовать все выводы микроконтроллера в проектируемом устройстве.



**Система прерываний** — одна из важнейших частей микроконтроллера. Особенность систем реального времени заключается в том, что для них чрезвычайно важным параметром является время реакции на внешние события. Поясним на простом примере. Когда вы производите математический расчет на компьютере, то обычно запускаете программу, предназначенную для выполнения этих расчетов, и после того, как она загрузится в память компьютера, вводите условие задачи и ждете результат. Время ожидания в таком случае не имеет принципиального значения (в пределах разумного, конечно) — медленная работа компьютера может раздражать, но на результате это не скажется. Система реального времени предполагает совершенно конкретную, рассчитываемую на этапе разработки скорость реакции системы управления на внешние события. Задержки сверх расчетов здесь просто недопустимы — они могут приводить к катастрофическим последствиям.

Проблемы быстрой реакции на события решаются организацией системы прерываний. При этом для каждого такого события разрабатывается отдельный «кусочек» кода, который формирует реакцию микроконтроллера на него. Этот «кусочек» кода называют подпрограммой обработки запроса на прерывание (для краткости часто используют термин **подпрограмма прерывания**) и размещают в памяти программ по известному адресу. В момент возникновения заданного события сигнал об этом поступает на вход **контроллера прерываний**, который представляет собой устройство, устанавливающее однозначное соответствие между входным сигналом о произошедшем событии и адресом программной памяти, по которому размещена точка входа в подпрограмму обработки запроса прерывания от данного события. Контроллер прерывает выполнение ЦП текущей программы и инициирует его переход на выполнение подпрограммы обработки прерывания. Время, прошедшее с момента возникновения события до начала выполнения первой инструкции подпрограммы прерывания, называют временем реакции микроконтроллера на событие. После окончания обработки ЦП автоматически возвращается к выполнению прерванной программы.

Другая функция контроллера прерываний — установка приоритетов событий. Понятие **приоритет** означает, что выполняемая подпрограмма прерывания может быть прервана другим событием только при условии, что оно имеет более высокий приоритет, чем текущее. В противном случае ЦП перейдет к обработке нового события после окончания обработки предыдущего. Контроллер прерываний, входящий в состав МК-8051, имеет пять входов событий: два от внешних устройств, два от таймеров и один от последовательного канала.

Обычно, когда говорят о каком-либо микроконтроллере, то всегда упоминают **семейство**, к которому он принадлежит. К одному семейству относят изделия, имеющие одинаковое **ядро**, под которым понимают совокупность таких понятий, как система команд, циклограмма работы ЦП, организация памяти программ и памяти данных, система прерываний и базовый набор периферийных устройств. Фактически на рис. 3.4 представлено ядро, ставшее основой для создания сотен других модификаций семейства МК-8051. Отличия между его различными представителями заключаются, в основном, в составе периферийных устройств и объеме памяти программ или данных. Поскольку диапазон задач, решаемых микроконтроллером, чрезвычайно широк, их производители стараются выпустить столько модификаций,

чтобы удовлетворить самые разнообразные запросы потребителей. Во многих семействах число модификаций приближается к сотне или даже превышает это значение.

Наиболее важная особенность семейства — программная совместимость на уровне двоичного кода всех входящих в него микроконтроллеров. Это позволяет разработчикам систем заменять одни микроконтроллеры семейства другими без потери наработок своего программного обеспечения. Естественно, чем большее число разновидностей входит в семейство, тем больше шансов выбрать оптимальный вариант, тем привлекательнее это семейство для разработчика. Вопрос правильного выбора семейства микроконтроллеров для новой разработки является стратегическим, так как проблема переноса программного обеспечения между изделиями разных семейств — чрезвычайно сложна, и даже использование языков высокого уровня не всегда позволяет решить ее без больших потерь. К вопросу о критериях выбора мы вернемся в следующих разделах книги.

## Кто изготавливает микроконтроллеры

Сегодня в мире выпускаются тысячи типов микроконтроллеров. В группе лидеров такие компании, как Atmel, Dallas Semiconductor, Intel, Infineon Technologies (бывшая Siemens Semiconductor Group), Hitachi Semiconductor, Microchip Technology Inc., Mitsubishi Electronics, Motorola Semiconductor, National Semiconductor, Oki Semiconductor, Philips Semiconductors, STMicroelectronics, Temic, Texas Instruments, Toshiba, Zilog и др.

Основным классификационным признаком микроконтроллеров является разрядность данных, обрабатываемых арифметико-логическим устройством (АЛУ). По этому признаку микроконтроллеры делятся на 4-, 8-, 16-, 32- и 64-разрядные.

Отдельно рассматриваются DSP-микроконтроллеры (DSP — Digital Signal Processor — цифровой сигнальный процессор), ориентированные на использование в системах обработки сигналов. Сегодня основная доля мирового рынка микроконтроллеров принадлежит восьмиразрядным устройствам (около 50% в стоимостном выражении). За ними следуют 16-разрядные и DSP-микроконтроллеры (каждая из групп занимает примерно по 20% рынка). Внутри каждой группы микроконтроллеры делятся на CISC- и RISC-устройства.

---

Совершенно закономерно, что и в России наиболее популярными стали именно восьмиразрядные микроконтроллеры. При этом исторически сложилось так, что электронная промышленность СССР в 80-е годы была в основном сориентирована на воспроизводство аналогов изделий фирмы Intel. Аналоги МК-8048 и МК-8051 выпускали предприятия в Минске, Киеве, Воронеже, Новосибирске, на них выросло целое поколение отечественных разработчиков.

---

Большое количество самой разнообразной информации по микроконтроллерам AVR фирмы ATMEL можно найти в Internet на следующих сайтах: [www.atmel.com](http://www.atmel.com); [www.atmel.ru](http://www.atmel.ru); [www.kirov.ru/~ua4nx/](http://www.kirov.ru/~ua4nx/); [www.inteltek.ru](http://www.inteltek.ru); [mselec.com](http://mselec.com); [www.chat.ru/~avreal/](http://www.chat.ru/~avreal/).

Информацию о покупке микроконтроллеров смотрите на таких сайтах: [www.efo.ru](http://www.efo.ru); [www.fulcrum.ru](http://www.fulcrum.ru).

## Микроконтроллеры AVR фирмы «ATMEL»

В течение нескольких последних лет в журнале «Радио» постоянно печаталась серия статей А. Долгого (г. Москва), в которых достаточно подробно описывался процесс программирования микроконтроллеров. Большая часть информации в этих статьях посвящена программированию микроконтроллеров PIC16XXXX фирмы *MICRICHIP*.

---

В этой книге основное внимание будет уделено описанию принципов работы микроконтроллеров AVR фирмы *ATMEL*.

---

Основанная в 1984 г. фирма *Atmel Corp.* (США) на данный момент является одним из признанных лидеров в области производства широкого спектра микросхем электронных компонентов: микросхем энергонезависимой памяти, микроконтроллеров общего назначения и микросхем программируемой логики.

Начиная с середины 90-х годов, фирма «Atmel» начала активно развивать новое направление в своей деятельности — производство высокопроизводительных 8-разрядных RISC-микроконтроллеров для встраиваемых приложений, объединенных общим названием AVR.

Микроконтроллеры AVR приобрели большую популярность, привлекая разработчиков достаточно выгодным соотношением таких показателей, как цена, быстродействие и энергопотребление. Кроме того, важными параметрами являются удобные режимы программирования, доступностью программно-аппаратных средств поддержки и широкая номенклатура выпускаемых кристаллов. Микроконтроллеры этой серии используются в автомобильной электронике, бытовой технике, сетевых картах и материнских платах компьютеров, в мобильных телефонах нового поколения и т.д.

В рамках единой базовой архитектуры AVR-микроконтроллеры подразделяются на три семейства.

- Tiny AVR — дешевые и довольно простые по конструкции микроконтроллеры в 8-выводном исполнении.
- Classic AVR — базовая линия микроконтроллеров.
- Mega AVR — микроконтроллеры для сложных приложений, требующих большого объема памяти программ и данных.

В этой главе очень коротко будут рассмотрены некоторые основные моменты конструкции микроконтроллеров семейства Classic AVR, как наиболее подходящие для радиолюбительского применения.

При этом все микроконтроллеры семейства поддерживают несколько режимов пониженного энергопотребления, имеют блок прерываний, сторожевой таймер и допускают программирование непосредственно в готовом устройстве через последовательный интерфейс SPI.

### Особенности микроконтроллеров семейства Classic AVR:

- ♦ производительность, приближающаяся к 1 MIPS/МГц;
- ♦ усовершенствованная AVR RISC архитектура;

- ♦ отдельные шины памяти команд и данных, 32 регистра общего назначения
- ♦ Flash ПЗУ программ, с возможностью внутрисистемного перепрограммирования и загрузки через SPI последовательный канал, 1000 циклов стирание/запись;
- ♦ EEPROM память данных, с возможностью внутрисистемной загрузки через SPI последовательный канал, 100000 циклов стирание/запись;
- ♦ блокировка режима программирования;
- ♦ встроенный аналоговый компаратор, сторожевой таймер, порты SPI и UART, таймеры/счетчики;
- ♦ полностью статические приборы, работающие при тактовой частоте от 0 Гц до 20 МГц;
- ♦ диапазон напряжений питания от 1,8 В до 6,0 В;
- ♦ режимы энергосбережения: пассивный (idle) и стоповый (power down).

### Общие сведения

AVR Classic — самая обширная производственная линия среди других Flash-микроконтроллеров корпорации Atmel. Она представила первый 8-разрядный Flash-микроконтроллер в 1993 году и с тех пор непрерывно совершенствует технологию. Фирма постоянно работает над совершенствованием своей продукции в следующих направлениях:

- ♦ в снижении удельного энергопотребления (мА/МГц);
- ♦ расширения диапазона питающих напряжений (до 1.8 В), что существенно для продления ресурса батарейных систем;
- ♦ увеличения быстродействия до 16 млн. операций в секунду;
- ♦ встройки в изделия реально-временных эмуляторов и отладчиков;
- ♦ реализации функции самопрограммирования;
- ♦ совершенствовании и расширении количества периферийных модулей;
- ♦ встройки специализированных устройств (радиочастотный передатчик, USB-контроллер, драйвер ЖКИ, программируемая логика, контроллер DVD, устройства защиты данных) и др.

---

Успех AVR-микроконтроллеров объясняется возможностью простого выполнения проекта с достижением необходимого результата в кратчайшие сроки, чему способствует доступность большого числа инструментальных средств проектирования, поставляемых, как непосредственно корпорацией Atmel, так и сторонними производителями. Ведущие сторонние производители выпускают полный спектр компиляторов, программаторов, ассемблеров, отладчиков, разъемов и адаптеров. Отличительной чертой инструментальных средств от Atmel является их невысокая стоимость.

---

Таким образом, AVR-микроконтроллеры представляют более широкие возможности по оптимизации производительности/энергопотребления, что особенно важно при разработке приложений с питанием от батареек. Микроконтроллеры обеспечивают производительность до 16 млн. операций в секунду и поддерживают Flash-память программ различной емкости: 1–256 кбайт.

В таблицах 3.1, 3.2 и 3.3 приведены некоторые данные по основным семействам AVR микроконтроллеров.

**Таблица 3.1.** Микроконтроллеры с архитектурой TinyAVR

Тип	Напр. питания, В	Такт. частота, МГц	I/O	FLASH	EEP-ROM	SRAM (ОЗУ)	АЦП	Корпус
ATtiny 11	2,7-5,5	6	6	1K	—	—	—	PDIP8 SOIC8
ATtiny12	1,8-5,5	6	6	1K	64	—	—	PDIP8 SOIC8
ATtiny13	1,8-5,5	20	6	1K	64	64	4x10bit	PDIP8 SOIC8
ATtiny15	2,7-5,5	6	6	1K	64	—	4x10bit	PDIP8 SOIC8
ATtiny2313	1,8-5,5	20	15	2K	128	128	—	PDIP20 SOIC20 MLF32
ATtiny26	2,7-5,5	16	16	1K	128	128	11x10 bit	PDIP20 SOIC20 MLF32
ATtiny28	1,8-5,5	4	20	2K	—	—	—	PDIP20 SOIC20 MLF32

**Таблица 3.2.** Микроконтроллеры с архитектурой Classic AVR

Тип	Напр. питания, В	Такт. частота, МГц	I/O	FLASH	EEP-ROM	SRAM (ОЗУ)	АЦП	Корпус
AT90S1200	2,7-6,0 4,0-6,0	4 12	15	1K	64	—	—	DIP20 SO20 SSOP20
AT90S2313	2,7-6,0 4,0-6,0	4 10	15	2K	128	128	—	DIP20 SO20
AT90LS2323	2,7-6,0	4	3	2K	128	128	—	DIP8 SO8
AT90S2323	4,0-6,0	10	3	2K	128	128	—	DIP8 SO8
AT90LS2343	2,7-6,0	4	5	2K	128	128	—	DIP8 SO8
AT90S2343	4,0-6,0	10	5	2K	128	128	—	DIP8 SO8
AT90LS4433	2,7-6,0	4	20	4K	256	128	6x10 bit	DIP28 TQFP3
AT90S4433	4,0-6,0	8	20	4K	256	128	6x10 bit	DIP28 TQFP3
AT90LS8515	2,7-6,0	4	32	8K	512	512	—	DIP40 TQFP4 PLCC44
AT90S8515	4,0-6,0	8	32	8K	512	512	—	DIP40 TQFP4 PLCC44

Таблица 3.2. Окончание

Тип	Напр. питан., В	Такт. частота, МГц	I/O	FLASH	EEP-ROM	SRAM (ОЗУ)	АЦП	Корпус
AT90LS8535	2,7-6,0	4	32	8K	512	512	8x10 bit	DIP40 TQFP4 PLCC44
AT90S8535	4,0-6,0	8	32	8K	512	512	8x10 bit	DIP40 TQFP4 PLCC44

Таблица 3.3. Микроконтроллеры с архитектурой MegaAVR

Тип	Напр. питан., В	Такт. частота, МГц	I/O	FLASH	EEP-ROM	SRAM (ОЗУ)	АЦП	Корпус
ATmega48	1,8-5,5	20	23	4K	256	512	6x10 bit 2x8bit	DIP28 TQFP3 MLF32
ATmega88	1,8-5,5	20	23	8K	512	1K	6x10 bit 2x8bit	DIP28 TQFP3 MLF32
ATmega168	1,8-5,5	20	23	16K	512	1K	6x10 bit 2x8bit	DIP28 TQFP3 MLF32
ATmega8	2,7-5,5	16	23	8K	512	1K	8x10 bit	DIP28 TQFP3 MLF32
ATmega16	2,7-5,5	16	32	16K	512	1K	8x10 bit	DIP40 TQFP4 MLF44
ATmega32	2,7-5,5	16	32	32K	1K	2K	8x10 bit	DIP40 TQFP4 MLF44
ATmega64	2,7-5,5	16	53	64K	2K	4K	8x10 bit	TQFP64 MLF64
ATmega128	2,7-5,5	16	53	128K	4K	4K	8x10 bit	TQFP64 MLF64
AT90CAN128	2,7-5,5	16	53	128K	4K	4K	8x10 bit	TQFP64 MLF64
ATmega103	4,0-5,5	8	48	128K	4K	4K	8x10 bit	TQFP64
ATmega161	2,7-5,5	8	35	16K	512	1K	—	DIP40 TQFP44 MLF44
ATmega162	1,8-5,5	16	35	16K	512	1K	—	DIP40 TQFP44 MLF44
ATmega163L	2,7-5,5	8	32	16K	512	1K	8x10 bit	DIP40 TQFP44 MLF44
ATmega169	1,8-5,5	4	534x25 LCD	16K	512	1K	8x10 bit	TQFP64
ATmega8515	2,7-5,5	16	35	8K	512	512	—	PDIP40 PLCC4 TQFP44

Таблица 3.3. Окончание

Тип	Напр. питан., В	Такт. частота, МГц	I/O	FLASH	EEP-ROM	SRAM (ОЗУ)	АЦП	Корпус
ATmega 8535	2,7-5,5	16	32	8K	512	512	8x10 bit	PDIP40 PLCC4 TQFP44
ATmega 325	1,8-5,5	16	53	32K	1K	2K	8x10 bit	TQFP MLF
ATmega 3250	1,8-5,5	16	68	32K	1K	2K	8x10 bit	TQFP MLF
ATmega 645	1,8-5,5	16	53	64K	2K	4K	8x10 bit	TQFP MLF
ATmega 6450	1,8-5,5	16	68	64K	2K	4K	8x10 bit	TQFP MLF

Все приборы одного семейства AVR совместимы по исходным кодам и тактированию. Семейство обеспечено комплектом программ и системами отладки, включающими: макро-ассемблеры, отладчики/симуляторы программ, внутрисхемные эмуляторы и отладочные устройства.

Микроконтроллеры семейства AVR поставляются в очищенном состоянии — содержимое и Flash-памяти программ и ЭСППЗУ данных находится в состоянии FF и готово к программированию.

### Характеристики ядра контроллера

Основными характеристиками центрального процессора микроконтроллеров рассматриваемого семейства Classic являются:

- ♦ полностью статическая архитектура; минимальная тактовая частота равна нулю;
- ♦ АЛУ подключено непосредственно к регистрам общего назначения;
- ♦ большинство команд выполняется за один машинный цикл;
- ♦ многоуровневая система прерываний;
- ♦ поддержка очереди прерываний;
- ♦ от 3 до 16 источников прерываний (из них до 2 внешних);
- ♦ наличие программного стека.

### Характеристики подсистемы ввода/вывода

Основными характеристиками подсистемы ввода/вывода являются:

- ♦ программное конфигурирование и выбор портов ввода/вывода;
- ♦ каждый вывод может быть запрограммирован как входной или как выходной независимо от других;
- ♦ входные буферы с триггером Шмитта на всех выводах;
- ♦ возможность подключения ко всем входам внутренних подтягивающих резисторов (сопротивление резисторов составляет 35–120 кОм);
- ♦ нагрузочная способность всех выводов составляет до 20 мА, что позволяет непосредственно управлять светодиодными индикаторами.

## Периферийные устройства

Микроконтроллеры семейства **Classic** обладают достаточно развитой периферией. Набор периферийных устройств, имеющих в составе того или иного микроконтроллера, зависит от конкретной модели. Перечислим все периферийные устройства, так или иначе встречающиеся в микроконтроллерах семейства:

- ♦ 8-разрядный таймер/счетчик с определителем (таймер T0);
- ♦ 16-разрядный таймер/счетчик с определителем (таймер T1);
- ♦ 8-разрядный таймер/счетчик с возможностью работы в асинхронном режиме (таймер T2);
- ♦ сторожевой таймер (WDT);
- ♦ одно- или двухканальный 8–10-разрядный генератор сигнала с широтно-импульсной модуляцией (ШИМ);
- ♦ одноканальный 8-разрядный генератор сигнала с ШИМ;
- ♦ аналоговый компаратор;
- ♦ 10-разрядный АЦП (6 или 8 каналов);
- ♦ полнодуплексный универсальный асинхронный приемопередатчик (UART);
- ♦ последовательный синхронный интерфейс SPI.

## Архитектура ядра

Ядро является «сердцем» микроконтроллеров AVR. Оно выполнено по усовершенствованной RISC (enhanced RISC) архитектуре, в которой используется ряд решений, направленных на повышение быстродействия микроконтроллеров. Схематическое упрощенное изображение ядра показано на рис. 3.5.

Арифметико-логическое устройство (АЛУ) выполняет все вычисления и непосредственно подключено к 32 рабочим регистрам, объединенным в регистровый файл. Благодаря этому АЛУ выполняет одну операцию (чтение содержимого регистров, выполнение операции и запись результата обратно в регистровый файл) за один машинный цикл.

В микроконтроллерах AVR практически все команды (за исключением команд, у которых одним из операндов является 16-разрядный адрес) занимают одну ячейку памяти программ.

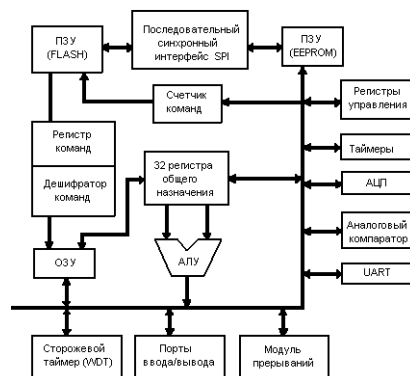


Рис. 3.5. Архитектура ядра микроконтроллера

## Цоколевка и описание выводов

Наиболее распространенное среди радиолюбителей семейство микроконтроллеров AVR Classic состоит в общей сложности из 17 моделей. Все они выпускаются в корпусах различных типов, что позволяет выбрать модель, наилучшим образом отвечающую требованиям по компоновке.



В табл. 3.2 приводятся основные параметры микроконтроллеров, такие как объем памяти (программ и данных), количество контактов ввода/вывода, тип корпуса, диапазон рабочих частот и напряжение питания.

Полную информацию по каждой модели следует искать в специальной литературе. Дополнительно следует отметить, что все микроконтроллеры семейства Classic выпускаются или на коммерческих (диапазон рабочих температур 0...+70°C), или на промышленных предприятиях (диапазон рабочих температур -40...+85°C).

## Микроконтроллеры модели AT90S1200

Далее рассмотрим только микроконтроллеры модели AT90S1200, которые являются типичными представителями семейства Classic AVR.

Кроме того, в приложении 4 приведено достаточно подробное описание еще одного очень популярного микроконтроллера — AT90S2313.

Данные по остальным моделям AVR микроконтроллеров можно найти в специальной литературе.

На рис. 3.6 показано расположение выводов микроконтроллера модели AT90S1200.

В табл. 3.4 приведены названия выводов для микроконтроллера AT90S1200 и указаны их функции (как основные, так и дополнительные). Кроме того, для каждого вывода в таблице указан его тип (вход, выход, вход/выход, вывод питания).

В табл. 3.4 использованы следующие обозначения выводов:

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

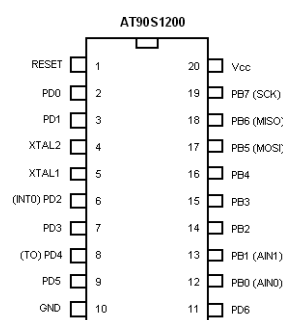


Рис. 3.6. Расположение выводов AT90S1200

Таблица 3.4. Описание выводов модели AT90S1200

Обозначение	Номер вывода	Тип вывода	Описание
XTAL1	5	I	Вход инвертора генератора и вход внешнего тактового сигнала
XTAL2	4	O	Выход инвертора генератора
RESET	1	I	Вход сброса. При удержании на входе НИЗКОГО уровня в течение 50 нс выполняется сброс устройства
<b>Порт В.</b> 8-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами			
PB0 (AIN0)	12	I/O	B0 (Положительный вход компаратора)
PB1 (AIN1)	13	I/O	B1 (Отрицательный вход компаратора)
PB2	14	I/O	B2
PB3	15	I/O	B3
PB4	16	I/O	B4
PB5 (MPS1)	17	I/O	B5 (Вход данных при последовательном программировании (SPI))

Таблица 3.4. Окончание

Обозначение	Номер вывода	Тип вывода	Описание
PB6 (MISO)	18	I/O	B6 (Выход данных при последовательном программировании (SPI))
PB7 (SCK)	19	I/O	B7 (Вход тактового сигнала при последовательном программировании SPI))
<b>Порт D. 7-разрядный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами</b>			
PD0	2	I/O	D0
PD1	3	I/O	D1
PD2 (INT0)	6	I/O	D2 (Вход внешнего прерывания)
PD3	7	I/O	D3
PD4 (T0)	8	I/O	D4 (Вход внешнего тактового сигнала таймера/счетчика T0)
PD5	9	I/O	D5
PD6	11	I/O	D6
GND	10	P	Общий вывод
Vcc	20	P	Вывод источника питания

Структурная схема микроконтроллера AT90S1200 приведена на рис. 3.7. Его отличительные особенности:

- 2 порта ввода/вывода: В (8-разрядный) и D (7-разрядный);

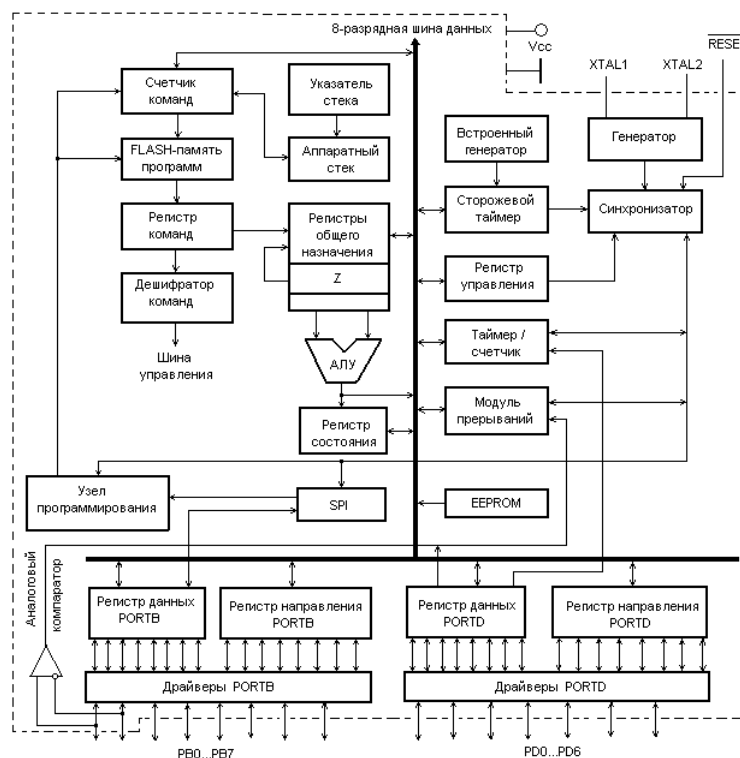


Рис. 3.7. Структурная схема микроконтроллера AT90S1200

- 3-уровневый аппаратный стек;
- встроенный тактовый RC-генератор;
- аналоговый компаратор;
- возможность подключения внешнего кварцевого резонатора.

## Организация памяти

Организация памяти микроконтроллеров AVR семейства Classic, в том числе и микроконтроллера AT90S1200, выполнена по гарвардской архитектуре, в которой разделены не только адресные пространства памяти программ и памяти данных, но также и шины доступа к ним. Причем память данных состоит из трех областей: регистровая память, статическое ОЗУ и память на основе EEPROM. В связи с тем, что регистровая память находится в адресном пространстве ОЗУ, об этих двух областях памяти обычно говорят как об одной. Каждая из областей (ОЗУ и EEPROM) расположена в своем адресном пространстве.

Отметим, что модель AT90S1200 не имеет внутреннего ОЗУ (хотя регистровая память, естественно, присутствует).

Обобщенная карта памяти микроконтроллеров AVR семейства Classic приведена на рис. 3.8.

Следует заметить, что:

- поскольку микроконтроллеры AVR имеют 16-разрядную систему команд, объем памяти программ на рисунке указан не в байтах, а в 16-разрядных словах;
- символ «\$» перед числом означает, что это число записано в шестнадцатеричной системе счисления.

**Память программ** предназначена для хранения команд, управляющих функционированием микроконтроллера. В памяти программ хранятся также различные константы, не меняющиеся во время работы программы. Память программ в микроконтроллерах семейства Classic представляет собой электрически стираемое ППЗУ (FLASH-ПЗУ). Поскольку все команды занимают в памяти 16 бит (некоторые — 32 бита), память программ имеет 16-разрядную организацию. Соответственно, объем памяти составляет от 512 16-разрядных слов для модели AT90S1200 до 40% слов для старших моделей.

Для адресации памяти программ используется счетчик команд (PC — Program Counter). Размер счетчика команд — от 9 до 12 разрядов в зависимости от объема адресуемой памяти.

По адресу \$000 памяти программ находится вектор сброса. После инициализации (сброса) микроконтроллера выполнение программы начинается с этого адреса (фирма Atmel рекомендует размещать по этому адресу команду относительного перехода к инициализационной части программы).

Начиная с адреса \$001, располагается таблица векторов прерывания. Ее размер зависит от модели микроконтроллера и составляет от 2 (адреса \$001, \$002) до 16

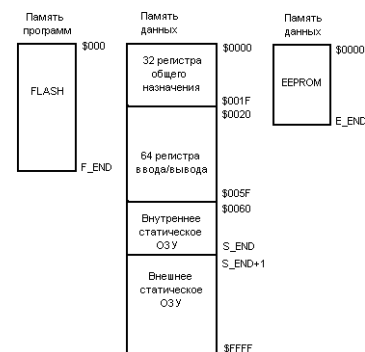


Рис. 3.8. Схема карты памяти микроконтроллеров семейства Classic

(адреса \$001–\$010) векторов. При возникновении прерывания после сохранения в стеке текущего значения счетчика команд происходит выполнение команды, расположенной по адресу соответствующего вектора. Поэтому по этим адресам располагаются команды относительного перехода к подпрограммам обработки прерываний. Ниже приведен типичный листинг начала программы для модели AT90S1200.

**Листинг 3.1.** Текст начала программы для AT90S1200

---

```

;-----
Address      Labels      Code      Comments
$000          rjmp RESRT    ; Обработчик сброса
$001          rjmp EXT INTO  ; Обработчик внешнего прерывания
$002          rkmp TIM OVFO  ; Обработчик прерывания
                        ; от таймера 0
$003      MAIN:      ldi r16,      ; Начало основной программы
                        low (RAMEND)
                        out SPL, r16
                        <инструкция> xxx
;-----

```

---

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

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

**Память данных** микроконтроллеров семейства Classic разделена на три части: регистровая память, оперативная память (статическое ОЗУ) и энергонезависимое ЭСППЗУ (EEPROM).

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

---

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

---

Для хранения переменных программ вместе с регистрами также может использоваться статическое ОЗУ объемом от 128 до 512 байт. Кроме того, микроконтроллеры AT90S4414 и AT90S8515 имеют возможность подключения внешнего статического ОЗУ объемом до 64 Кбайт.

Для хранения данных, которые могут изменяться в процессе настройки и функционирования готовой системы (калибровочные константы, серийные номера, ключи и т.п.), может быть использована EEPROM-память. Ее объем составляет для различных моделей от 64 до 512 байт. Эта память расположена в отдельном адресном пространстве, а доступ к ней осуществляется с помощью определенных РВВ.

## Статическое ОЗУ

В микроконтроллерах AVR семейства Classic используется линейная организация памяти. Объем статического ОЗУ для различных моделей семейства составляет от 128 до 512 байт (см. табл. 3.2).

В адресном пространстве ОЗУ также расположены все регистры микроконтроллеров, под них отведены младшие 96 адресов (см. рис. 3.9). Основные адреса отведены под 128/256/512...64 Кбайт ячеек статического ОЗУ.

На рис. 3.9 схематически показана организация статического ОЗУ.

## Регистры общего назначения

Прежде чем перейти к описанию имеющихся в микроконтроллере регистров, хочу обратить ваше внимание на то, что в приложении 4 все регистры микроконтроллеров AT90S2312 рассмотрены очень подробно. Поскольку большинство регистров AT90S1200 идентичны описанным в приложении 4, далее в этой главе опишем только некоторых общие их особенности.

Все регистры общего назначения (РОН) объединены в файл, структура которого показана на рис. 3.10.

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

Ряд регистров общего назначения используется в качестве указателей при косвенной адресации памяти данных. В модели AT90S1200 таким регистром является регистр R30 (регистр Z). Поскольку объем адресуемой памяти данных этой модели составляет всего 96 байт, для хранения адреса достаточно одного 8-разрядного регистра. Во всех других моделях для косвенной адресации используются три 16-разрядных регистра (регистры X, Y и Z),

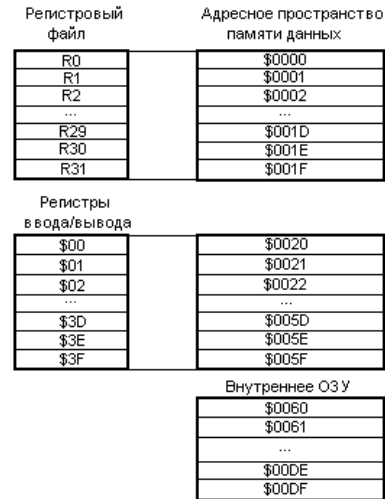


Рис. 3.9. Схема организации статического ОЗУ

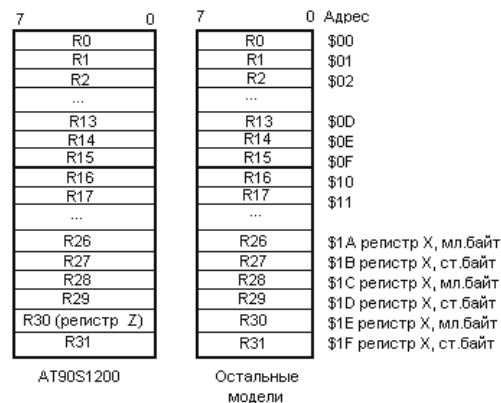


Рис. 3.10. Структура регистров общего назначения

каждый из которых получается объединением двух РОН (рис. 3.11).

Как показано на рис. 3.10, каждый регистр файла имеет свой собственный адрес в пространстве памяти данных (кроме AT90S1200). Поэтому к ним можно обращаться как к памяти, несмотря на то, что физически эти регистры не являются ячейками ОЗК.

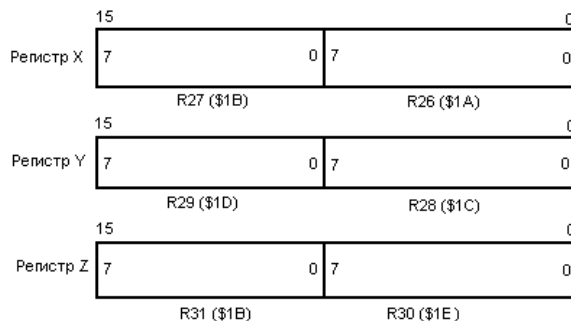


Рис. 3.11. Регистры-указатели X, Y и Z

### Регистры ввода/вывода

Регистры ввода/вывода (PBB) располагаются в так называемом пространстве ввода/вывода размером 64 байта. Все PBB можно разделить на две группы:

- ♦ служебные регистры микроконтроллера;
- ♦ регистры, относящиеся к периферийным устройствам (в т.ч. порты ввода/вывода).

Размер каждого регистра — 8 разрядов.

Распределение адресов пространства ввода/вывода зависит от конкретной модели микроконтроллера, т.к. разные модели имеют различный состав периферийных устройств и, соответственно, разное количество регистров.

К любому регистру ввода/вывода можно обратиться с помощью команд IN и OUT, выполняющих пересылку данных между одним из 32 РОН и пространством ввода/вывода. Кроме того, имеются 4 команды поразрядного допуска, использующие в качестве операндов регистры ввода/вывода (табл. 3.5) команды установки/сброса отдельного разряда (SBI и CBI) и команды проверки состояния отдельного разряда (SBIS и SBIC).

Таблица 3.5. Регистры ввода/вывода модели AT90S1200 и AT90S2313

Название	Функция	AT90S1200	AT90S2313
ACSR	Регистр управления и состояния аналогового компаратора	\$08	\$08 (\$28)
DDRB	Регистр направления данных порта B	\$17	\$17 (\$37)
DDRD	Регистр направления данных порта D	\$11	\$11 (\$31)
EEAR	Регистр адреса EEPROM	\$IE	\$IE (\$3E)
EECR	Регистр данных EEPROM	\$IC	\$IC (\$3C)
EEDR	Общий регистр флагов прерываний	\$ID	\$ID (\$3D)
GIFR	Общий регистр маски прерываний	—	\$3A (\$5A)
GIMSK	Регистр захвата таймера/счетчика 1 (старший байт)	\$3B	\$3B (\$5B)
ICR1H	Регистр захвата таймера/счетчика 1 (младший байт)	—	\$25 (\$45)
ICR1L	Общий регистр управления микроконтроллером	—	\$24 (\$44)
MCUCR	Регистр состояния микроконтроллера	\$35	\$35 (\$55)
MCUSR	Регистр состояния микроконтроллера	—	—
OCR1AH	Регистр совпадения выхода 1 (старший байт)	—	\$2B (\$4B)

Таблицы 3.5. Окончание

Название	Функция	AT90S1200	AT90S2313
OCRIAL	Регистр совпадения выхода 1 (младший байт)	—	\$2A (\$4A)
PINB	Выводы порта B	\$16	\$16 (\$36)
PIND	Выводы порта D	\$10	\$10 (\$30)
PORTB	Регистр данных порта B	\$18	\$18 (\$38)
PORTD	Регистр данных порта D	\$12	\$12 (\$32)
SPL	Указатель стека	—	\$3D (\$5D)
SREG	Регистр состояния	\$3F	\$3F (\$5F)
TCCR0	Регистр управления таймером/счетчиком 0	\$33	\$33 (\$53)
TCCRIA	Регистр управления A таймером/счетчиком 1	—	\$2F (\$4F)
TCCRIB	Регистр управления B таймером/счетчиком 1	—	\$2E (\$4E)
TCNTIL	Счетный регистр таймера/счетчика 1 (младший байт)	—	\$2C (\$4C)
TCNT0	Счетный регистр таймера/счетчика 0 (8-разрядный)	\$32	\$32 (\$52)
TCNTIH	Счетный регистр таймера/счетчика 1	—	\$2D (\$4D)
TIFR	Регистр флагов прерываний от таймера/счетчика	\$38	\$38 (\$58)
TIMSK	Регистр маски прерываний от таймера/счетчика	\$39	\$39 (\$59)
UBRR	Регистр скорости передачи UART	—	\$09 (\$29)
UCR	Регистр управления UART	—	\$0A (\$2A)
UDR	регистр данных UART	—	\$0C (\$2C)
USR	Регистр состояния UART	—	\$0B (\$2B)
WDTCR	Регистр управления сторожевым таймером	\$21	\$21 (\$41)

### Служебные регистры микроконтроллера

Далее рассмотрим служебные регистры микроконтроллера. Обратите внимание, что адреса служебных регистров не меняются от моделей к модели, т.е. регистр SREG всегда расположен по адресу \$3F (\$5F), регистр TIMSK — по адресу \$3B (\$5B) и т.д.

**SREG** (регистр состояния). Регистр состояния располагается по адресу \$3F (\$5F). Этот регистр представляет собой набор флагов, показывающих текущее состояние микроконтроллера. Эти флаги автоматически устанавливаются в «1» или в «0» при наступлении определенных событий (в соответствии с результатом выполнения команд). Все разряды этого регистра доступны как для чтения, так и для записи в любой момент времени; после сброса микроконтроллера все разряды регистра сбрасываются в «0».

**SP** (указатель стека) — в моделях, имеющих объем до 128 байт (адресное пространство ОЗУ — \$000–\$0DF), указатель стека реализован на одном регистре SPL, расположенном по адресу \$3D (\$5D). В остальных моделях указатель стека реализован на паре регистров SPH:SPL, соответственно расположенных по адресам \$3E (\$5E) и \$3D (\$5D).

---

Все используемые разряды регистров доступны как для чтения, так и для записи в любой момент времени. После сброса микроконтроллера содержимое регистров равно 0, поэтому в самом начале программы указатель стека необходимо проинициализировать каким-либо значением (как правило, это наибольший для конкретного микроконтроллера адрес памяти данных).

---

**GIMSK, TIMSK, GIFR, TIFR** (регистры управления прерываниями) — эти четыре регистра предназначены для управления внешними прерываниями (регистры GIMSK и GIFR) и прерываниями от таймеров (регистры TIMSK и GIFR) и прерываниями от таймеров (регистры TIMSK и TIFR). Регистры масок GIMSK (общий регистр маски прерываний) и TIMSK (регистр маски прерываний от таймеров) используются для разрешения/запрещения отдельных прерываний, а регистры флагов GIFR (общий регистр флагов прерываний) и TIFR (регистр флагов прерываний от таймеров) содержат флаги, показывающие, произошло или нет соответствующее прерывание.

**MCUCR** (регистр управления микроконтроллером) — расположен по адресу \$35 (\$55). Этот регистр содержит ряд флагов, используемых для общего управления микроконтроллером. Состав флагов, размещенных в регистре MCUCR, несколько меняется от модели к модели, соответственно в определенных моделях некоторые разряды не используются. Неиспользуемые разряды регистра доступны только для чтения и содержат «0». Все используемые разряды регистра доступны как для чтения, так и для записи в любой момент времени. После сброса микроконтроллера во всех разрядах регистра записано «0».

**MCUSR** (регистр состояния микроконтроллера) — расположен по адресу \$34 (\$54). Этот регистр содержит флаги, состояние которых позволяет определить причину, по которой произошел сброс микроконтроллера.

### Способы адресации памяти данных

Все микроконтроллеры AVR семейства Classic, за исключением модели AT90S1200, поддерживают 8 способов адресации для доступа к различным областям памяти данных (РОН, РВВ, ОЗУ). Модель AT90S1200 в связи с отсутствием у нее встроенного ОЗУ и из-за наличия единственного индексного регистра поддерживает только 4 способа адресации из восьми.

В действительности способов адресации всего два:

- ♦ прямая адресация;
- ♦ косвенная адресация.

Однако каждый способ адресации имеет несколько разновидностей в зависимости от того, к какой области памяти производится обращение (для прямой адресации) или какие дополнительные действия выполняются над индексным регистром (для косвенной адресации).

#### Прямая адресация

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

#### Прямая адресация одного регистра общего назначения

Этот способ адресации используется в командах, оперирующих с одним из регистров общего назначения. При этом адрес регистра-операнда (его номер) содержится в разрядах 8...4 (5 бит) слова команды.



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

### **Прямая адресация двух регистров общего назначения**

Этот способ адресации используется в командах, оперирующих одновременно с двумя регистрами общего назначения. При этом адрес регистра-источника содержится в разрядах 9, 3...0 (5 бит), а адрес регистра-приемника в разрядах 8...4 (5 бит) слова команды.

К командам, использующим этот способ адресации, относится команда пересылки данных из регистра в регистр (MOV), а также большинство команд арифметических операций.

### **Прямая адресация регистра ввода/вывода**

Данный способ адресации используется командами пересылки данных между регистром ввода/вывода и регистровым файлом — IN и OUT. В этом случае адрес регистра ввода/вывода содержится в разрядах 10, 9, 3...0 (6 бит), а адрес РОН — в разрядах 8...4 (5 бит).

### **Прямая адресация ОЗУ**

Как следует из названия, данный способ используется при обращении ко всему адресному пространству памяти данных. Естественно, этот способ адресации не поддерживается микроконтроллером AT90S1200.

В системе команд микроконтроллеров семейства имеется только две команды, использующие этот способ адресации. Это команды пересылки байта между одним из РОН и ячейкой ОЗУ — LDS и STS. Каждая из этих команд занимает в памяти программ два слова (32 бита). В первом слове содержится код операции и адрес регистра общего назначения (в разрядах с 8-го по 4-й). Во втором слове находится адрес ячейки памяти, к которой происходит обращение.

---

Еще раз обращаем ваше внимание, что по адресам \$00–\$1F расположен файл регистров общего назначения, а по адресам \$20–\$5F расположены регистры ввода/вывода.

---

### **Косвенная адресация**

При косвенной адресации адрес ячейки памяти (для AT90S1200 — регистра) находится в одном из индексных регистров X, Y и Z. В зависимости от дополнительных манипуляций, которые производятся над содержимым индексного регистра, различают следующие разновидности косвенной адресации: простая косвенная адресация, относительная косвенная адресация, косвенная адресация с преддекрементом и косвенная адресация с постинкрементом.

#### **Простая косвенная адресация**

Сразу отметим, что AT90S1200 поддерживает только этот вид косвенной адресации. При использовании команд простой косвенной адресации обращение производится по адресу (регистра — для AT90S1200, ячейки памяти — для остальных моделей), который находится в индексном регистре. Никаких действий с содержимым индексного регистра при этом не производится.

Микроконтроллеры поддерживают 6 команд (по 2 для каждого индексного регистра) простой косвенной адресации:  $LD\ Rd, X/Y/Z, Rd$  (пересылка байта из РОН в ОЗУ). Адрес регистра общего назначения содержится в разрядах 8...4 слова команды.

### Относительная косвенная адресация

При использовании команд относительной косвенной адресации адрес ячейки памяти, к которой обращаются, получается суммированием содержимого индексного регистра (Y или Z) и константы, задаваемой в команде. Другими словами, производится обращение по адресу, указанному в команде, относительно адреса, находящегося в индексном регистре.

Соответственно микроконтроллеры поддерживают 4 команды относительной косвенной адресации (две для регистра Y и две для регистра Z):  $LDD\ Rd, Y+q/Z+q$  (пересылка байта из ОЗУ и РОН) и  $ST\ Y+q/Z+q, Rr$  (пересылка байта из РОН в ОЗУ). Адрес регистра общего назначения содержится в разрядах 8...4 слова команды, а величина смещения — разрядах 13, 11, 10...0. Поскольку под значение смещения отводится только 6 бит, оно не может превышать 64.

### Косвенная адресация с преддекрементом

При использовании команд косвенной адресации с преддекрементом содержимое индексного регистра сначала увеличивается на 1, а затем производится обращение по полученному адресу. Микроконтроллеры семейства поддерживают 6 команд (по 2 для каждого индексного регистра) косвенной адресации с преддекрементом:  $LD\ Rd, -X/-Y/-Z$  (пересылка байта из РОН в ОЗУ). Адрес регистра общего назначения содержится в разрядах 8...4 слова.

### Косвенная адресация с постинкрементом

При использовании команд косвенной адресации с постинкрементом после обращения по адресу, который находится в индексном регистре, содержимое индексного регистра уменьшается на 1.

Микроконтроллеры семейства поддерживают 6 команд (по 2 для каждого индексного регистра) косвенной адресации с постинкрементом:  $LD\ Rd, X+/Y+/Z+$  (пересылка байта из ОЗУ в РОН) и  $ST\ X+/Y+/Z+, Rd$  (пересылка байта из РОН в ОЗУ). Адрес регистра общего назначения содержится в разрядах 8...4 слова команды.

### Энергонезависимая память данных

Как уже было сказано, микроконтроллеры AVR семейства Classic имеют в своем составе энергонезависимую память (EEPROM). Объем этой памяти колеблется от 64 байт в модели AT90S1200 до 512 байт в старших моделях. EEPROM-память расположена в своем адресном пространстве и так же, как и ОЗУ, организована линейно.

### Организация доступа

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

## Регистр адреса

В регистр адреса загружается адрес ячейки, к которой будет производиться обращение. В моделях микроконтроллеров с объемом EEPROM-памяти до 256 байт регистр адреса реализован на одном PVB (регистр ввода/вывода) — EEAR (EEPROM Address Register), расположенном по адресу \$IE (\$3E). В моделях микроконтроллеров с объемом EEPROM-памяти, равным 512 байт, для адресации всего адресного пространства требуется уже девять разрядов, поэтому регистр адреса в них реализован на двух PVB — EEARN (старший байт адреса) и EEARL (младший байт адреса). Эти регистры расположены по адресам \$IF (\$3F) и \$IE (\$3E) соответственно.

Все перечисленные регистры доступны как для записи, так и для чтения. При этом содержимое разрядов 7...1 регистра EEARN, разумеется, игнорируется.

## Регистр данных

Регистр ввода/вывода, являющийся регистром данных EEPROM-памяти, называется EEDR (EEPROM Data Register), а расположен он по адресу \$ID (\$3D). При записи в этот регистр загружаются данные, которые должны быть помещены в EEPROM по адресу, находящемуся в регистре EEAR.

## Регистр управления

Как следует из названия, данный регистр используется для управления доступом к EEPROM-памяти. Этот регистр, который называется EECR (EEPROM Control Register), расположен по адресу \$IC (\$3C). Разные модели предоставляют неодинаковые возможности по управлению процессами записи/чтения в EEPROM, поэтому состав управляющих разрядов в регистре EECR зависит от конкретной модели микроконтроллера.

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

1. Дождаться готовности EEPROM к приему новых данных (ждать, пока не сбросится флаг EEWЕ регистра EECR).
2. Загрузить байт данных в регистр EEDR, а требуемый адрес — в регистр EEAR (EEARN:EEARL).
3. Установить в «1» флаг EEMWE регистра EECR. Причем для выполнения этой операции необходимо в том же машинном цикле записать «0» в разряд EEWЕ.
4. В течение 4 машинных циклов после установки флага EEMWE записать в разряд EEWЕ регистра EECR лог: «1».

Для микроконтроллера AT90S1200, в регистре EECR которого отсутствует флаг EEMWE, пункты 3 и 4 сводятся к простой установке разряда EEWЕ. Обратите внимание, что после установки этого разряда в «1» процессор пропускает 2 машинных цикла перед началом выполнения следующей инструкции.

Длительность цикла записи составляет 2–4 мс, в зависимости от напряжения питания микроконтроллера (2 мс при  $V_{cc} = 2,7$  В). По окончании цикла записи, разряд EEWЕ аппаратно сбрасывается, после чего программа может начать запись следующего байта.

С учетом сказанного фрагмент программы, осуществляющей запись в EEPROM, выглядит следующим образом (на примере модели AT90S1200):

```

;-----
EEWrite:
sbic      EECR, EEWE      ; ждать, пока флаг EEWE не будет
rjmp      EEWwrite       ; сброшен
sli       ; запретить прерывания
out       EEAR, AddrReq   ; загрузить адрес (AddrReq – POH)
out       EEAR, DataReq   ; загрузить данные (DataReq – POH)
sbi       EECR, EEWE      ; выдать строб записи в EEPROM
sli       ; разрешить прерывание (если необходимо)
;-----

```

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

С учетом сказанного фрагмент программы, осуществляющей чтение из EEPROM, выглядит следующим образом (на примере модели AT90S1200):

```

;-----
sbic      EECR, EEWE      ; ждать окончания текущей записи
rjmp      EERead         ; (пока флаг EEWE не станет равным "0")

;
out       EEDR, AddrReq   ; загрузить адрес (AddrReq -- POH)
sbi       EECR, EERE      ; выдать строб чтения из EEPROM
in        DataReq, EEDR   ; прочитанный байт – в POH DataReq
;-----

```

### Меры предосторожности при работе с EEPROM

К сожалению, у EEPROM-памяти есть один недостаток: во время работы при пониженном напряжении питания хранящиеся в ней данные могут быть повреждены. Это может произойти по двум следующим причинам.

1. Обычная процедура записи в EEPROM требует некоторого минимального напряжения питания; если напряжение питания ниже этой величины, запись не может быть выполнена.
2. Микроконтроллер сам может выполнять команды некорректно, если напряжение питания будет ниже определенной величины.

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

1. Удерживать микроконтроллер в состоянии сброса все время, пока напряжение питания находится ниже нормы. Это решение реализуется внешней схемой защитного сброса, называемой также детектором пониженного напряжения питания (Brown-out Detector).
2. Удерживать микроконтроллер в «спящем» режиме (Power Down), пока напряжение питания находится ниже нормы. Поскольку в этом режиме микроконтроллер не может выполнять никаких команд, такое решение эффективно защищает служебные регистры EEPROM от непреднамеренной записи.

3. Хранить константы во FLASH-памяти программ, если они не должны меняться во время работы программы. Микроконтроллер не может самостоятельно производить запись в FLASH-память, соответственно, при понижении напряжения питания ее содержимое не будет повреждено.

## Создание программ для микроконтроллера

Программа для любого микроконтроллера представляет собой последовательность команд, записанных в памяти программ. Большинство команд при выполнении изменяют содержимое одного или нескольких регистров общего назначения, регистров ввода/вывода или ячеек ОЗУ.

В конце приложения 4 располагаются табл. П4.29 и табл. П4.30, в которых перечислены все команды, которые могут применяться с микроконтроллерах фирмы Atmel.

Далее рассмотрим несколько проверенных на практике программ для микроконтроллеров AT90S1200 и AT90S2313, разработанных радиолюбителями. Все эти программы являются доступными и располагаются на различных сайтах в Internet.

## Программа для радиомаяка

В радиолюбительской практике часто встречается необходимость включать на определенной радиочастоте так называемый «радиомаяк». Это специальный мало-мощный радиопередатчик, передающий через какие-то промежутки времени одну и ту же информацию, в которой, как правило, содержится позывной радиостанции-маяка и координаты её местонахождения. Маяки обычно применяются для контроля прохождения радиоволн в том или ином направлении.

На рис. 3.12 показана принципиальная электрическая схема соединительных линий микроконтроллера, задействованного в конструкции радиомаяка.

После подачи напряжения манипулятор маяка начинает передавать записанную в него информацию азбукой Морзе. На выходах появляется аналогичная информация.

1. CW-key (прямой телеграфный выход) — для включения передатчика.

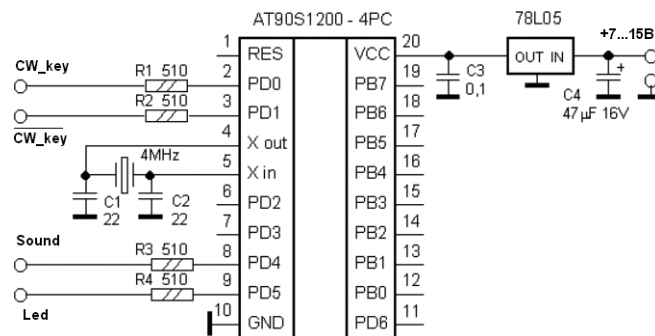


Рис. 3.12. Схема управления маяком

2. CW-key (инверсный телеграфный выход) — для выключения передатчика.
3. SOUND — подключение динамика или УНЧ.

4. LED — подключение светодиода на корпус, ток до 10 мА задается R4.

Общее потребление — 10–15 мА.

Микросхему AT90S1200 — 4PC можно заменить на AT90S1200 — 4PI, AT90S1200 — 12PC, AT90S1200 — 12PI. При необходимости больших объемов можно использовать AT90S2313 — 4PC, AT90S2313 — 4PI, AT90S2313 — 10PC, AT90S2313 — 10PI.

Далее рассмотрим листинги исходных кодов программы, предназначенной для управления микроконтроллера.

Листинг 3.2 — начало программы, выполненной на языке Ассемблера для микроконтроллеров фирмы Atmel.

#### Листинг 3.2. Файл UA4NM.asm

```
;*****
.DEVICE AT90S1200
;
; АВТОМАТ МОРЗЕ ( UA4NM )
;
; ПОРТЫ ВЫВОДА SW прямой — P3.0 , инверсный — P3.1",
; Светодиод — P3.5 , Звук — P3.4
; Кварц — 4 МГц
;
.CSEG; СЕГМЕНТ ПРОГРАММЫ
;-----
        LDI R20,0x0F
        OUT 0x21,R20      ; Включение СТОРОЖЕВОГО ТАЙМЕРА 2 секунды
        CLR R7            ; Очистить R7
        OUT 0x3B,R7      ; Общий регистр маски прерываний
        OUT 0x8,R7        ; Регистр упр. и сост. аналогового компаратора
        OUT 0x17,R7      ; Порт P1.0 — P1.7 -> Ввод
        LDI R20,0xFF
        OUT 0x11,R20     ; Порт P3.0–P3.7 -> Вывод
        OUT 0x12,R7      ; Порт D сброшен
;*****
```

В листинге 3.2 показаны исходные коды начала программы. В первой строке (.DEVICE AT90S1200) указана модель задействованного микроконтроллера. В следующей строке определен сегмент программы. Эти две строки предназначены для настройки компилятора.

В последующих строках кодов определены исходные данные.

Листинг 3.3 начинается с описания кодов, которые создают на выходе звуковой сигнал. Строка «Tone UA4NM QRA LO48UM TX 3 watt Ant GP» представляет собой передаваемый радиомаяком сигнал. Слово «Tone» говорит о том, что сигнал маяка начинается с выдачи в эфир монотонного сигнала в течение нескольких секунд. За-

```
; *****  
; ОСНОВНАЯ ПРОГРАММА  
; *****  
; Tone UA4NM QRA LO48UM TX 3 watt Ant GP  
;  
beacon: SBI 0x12,0 ; Установка Port P3.0 в "1"  
        CBI 0x12,1 ; Установка Port P3.1 в "0"  
        SBI 0x12,5 ; Установка Port P3.5 в "1"  
        LDI R20,0x70 ; Длительность тона  
T2:     rcall SOUND  
        DEC R20  
        BRNE T2 ; Повтор 32 раз.  
        SBI 0x12,1 ; Установка Port P3.1 в "1"  
        CBI 0x12,0 ; Установка Port P3.0 в "0"  
        CBI 0x12,5 ; Установка Port P3.5 в "0"  
START:  RCALL PAUSA3 ; Пауза после передачи монотонного сигнала  
        RCALL PAUSA2 ; Еще одна пауза  
        RCALL tochka ; U — начинает передаваться буква U как ..-  
        RCALL tochka  
        RCALL tire  
        RCALL PAUSA1 ; Пауза перед передачей второй буквы  
        RCALL tochka ; A — начинает передаваться буква A как .-  
        RCALL tire  
        RCALL PAUSA1 ; Пауза перед следующей буквой (и т.д.)  
        RCALL tochka ; 4  
        RCALL tochka | RCALL tire  
        RCALL tochka | RCALL tochka  
        RCALL tochka | RCALL tire  
        RCALL tire | RCALL PAUSA1  
        RCALL PAUSA1 | RCALL tire ; T  
        RCALL tire ; N | RCALL PAUSA1  
        RCALL tochka | RCALL tochka ; H  
        RCALL PAUSA1 | RCALL tochka  
        RCALL tire ; M | RCALL tochka  
        RCALL tire | RCALL tochka  
        RCALL PAUSA2 | RCALL PAUSA2  
        RCALL PAUSA3 | RCALL tochka ; I  
        RCALL tire ; O | RCALL tochka
```

```

RCALL PAUSA1
RCALL tochka      ; S
RCALL tochka
RCALL tochka
RCALL PAUSA2
RCALL tire        ; K
RCALL tochka
RCALL tire
RCALL PAUSA1
RCALL tochka      ; I
RCALL tochka
RCALL PAUSA1
RCALL tochka      ; R
RCALL tire
RCALL tochka
RCALL PAUSA1
RCALL tire        ; O
RCALL tire
RCALL tire
RCALL PAUSA1
RCALL tochka      ; V
RCALL tochka
RCALL tochka
RCALL tire
RCALL PAUSA3
RCALL tire        ; Q
RCALL tire
RCALL tochka
RCALL tire
RCALL PAUSA1
RCALL tochka      ; R
RCALL tire
RCALL tochka
RCALL PAUSA1
RCALL tochka      ; A
RCALL tire
RCALL PAUSA2
RCALL tochka      ; I
RCALL tochka
RCALL PAUSA1
RCALL tochka      ; S
RCALL tochka
RCALL tochka
RCALL PAUSA2

```

```

RCALL tochka      ; L
RCALL tire
RCALL tochka
RCALL tochka
RCALL PAUSA1
RCALL tire        ; O
RCALL tire
RCALL tire
RCALL PAUSA1
RCALL tochka      ; 4
RCALL tochka
RCALL tochka
RCALL tochka
RCALL tire
RCALL PAUSA1
RCALL tire        ; 8
RCALL tire
RCALL tire
RCALL tochka
RCALL tochka
RCALL PAUSA1
RCALL tochka      ; U
RCALL tochka
RCALL tire
RCALL PAUSA1
RCALL tire        ; O
RCALL tire
RCALL tire
RCALL PAUSA3
RCALL tire        ; T
RCALL PAUSA1
RCALL tire        ; X
RCALL tochka
RCALL tochka
RCALL tire
RCALL PAUSA2
RCALL tochka      ; 2
RCALL tochka
RCALL tire
RCALL tire
RCALL tire
RCALL PAUSA2
RCALL tochka      ; W
RCALL tire

```



RCALL tire		RCALL PAUSA3	
RCALL PAUSA1		RCALL tochka	; A
RCALL tochka	; A	RCALL tire	
RCALL tire		RCALL tire	; T
RCALL PAUSA1		RCALL PAUSA2	
RCALL tire	; T	RCALL tire	; G
RCALL PAUSA1		RCALL tire	
RCALL tire	; T	RCALL tochka	
RCALL PAUSA1		RCALL PAUSA1	
RCALL tochka	; S	RCALL tochka	; P
RCALL tochka		RCALL tire	
RCALL tochka		RCALL tire	
RCALL PAUSA1		RCALL tochka	
RCALL tire	; N	RCALL PAUSA3	
RCALL tochka		rjmp BEACON	
RCALL PAUSA1			

;\*\*\*\*\*

В листинге 3.3 расписаны подряд все выдаваемые в выходной порт телеграфные символы радиомаяка. В листинге 3.4 описаны все подпрограммы, которые вызываются для работы в предыдущем листинге.

#### Листинг 3.4. Продолжение2 файла UA4NM.asm

```
;*****
; Подпрограммы ТИРЕ, ТОЧКА, ПАУЗА, ПАУЗА1, ПАУЗА2, ЗВУК
;
;*****
tochka:                ; Первая подпрограмма — «Телеграфная точка»
    SBI 0x12,0          ; Установка Port P3.0 в "1"
    CBI 0x12,1          ; Установка Port P3.1 в "0"
    SBI 0x12,5          ; Установка Port P3.5 в "1"
    rcall SOUND
    SBI 0x12,1          ; Установка Port P3.1 в "1"
    CBI 0x12,0          ; Установка Port P3.0 в "0"
    CBI 0x12,5          ; Установка Port P3.5 в "0"
    rcall PAUSA
    RET

;-----
tire:                  ; подпрограмма телеграфного тире
    SBI 0x12,0          ; Установка Port P3.0 в "1"
    CBI 0x12,1          ; Установка Port P3.1 в "0"
    SBI 0x12,5          ; Установка Port P3.5 в "1"
    rcall SOUND
```

```

        rcall SOUND
        rcall SOUND
        LDI R19,0x27      ; ДЛИТЕЛЬНОСТЬ
        rcall S1
        SBI 0x12,1        ; Установка Port P3.1 в "1"
        CBI 0x12,0        ; Установка Port P3.0 в "0"
        CBI 0x12,5        ; Установка Port P3.5 в "0"
        rcall PAUSA
        RET

;-----
pausa:
        LDI R19,0x80      ; ДЛИТЕЛЬНОСТЬ ПАУЗЫ
P1:      LDI R21,0xFF
P2:      WDR
        WDR
        WDR
        DEC R21
        BRNE P2           ; Повтор FF раз.
        LDI R21,0xFF
P3:      WDR
        WDR
        WDR
        DEC R21
        BRNE P3           ; Повтор FF раз.
        DEC R19
        BRNE P1           ; Повтор FF раз.
        RET

;-----
PAUSA1: LDI R19,0xFF      ; ДЛИТЕЛЬНОСТЬ 2 точки
        rjmp P1           ; ПАУЗА между знаками с группе ( 2 + 1 точки )
;-----
PAUSA2: LDI R19,0xFF      ; ДЛИТЕЛЬНОСТЬ 2 точки
        RCALL P1          ; ПАУЗА между ГРУППАМИ ( 6 + 1 точки )
        LDI R19,0xFF      ; ДЛИТЕЛЬНОСТЬ 2 точки
        RCALL P1
        RET

;-----
PAUSA3:
        RCALL pausa2      ; ПАУЗА между предложениями (12+1 точки)
        RCALL pausa2
        RET

;-----
SOUND:  LDI R19,0x70      ; ДЛИТЕЛЬНОСТЬ ЗВУЧАНИЯ

```

```

S1:      LDI R21,0xFF
S2:      WDR
          SBI 0x12,4          ; Установка Port P3.4 в "1"
          DEC R21
          BRNE S2             ; Повтор FF раз.
          LDI R21,0xFF
S3:      WDR
          CBI 0x12,4          ; Установка Port P3.4 в "0"
          DEC R21
          BRNE S3             ; Повтор FF раз.
          DEC R19
          BRNE S1             ; Повтор FF раз.
          ret

```

Готовить тексты файла с расширением ASM можно в самом простом текстовом редакторе типа Notepad или в любом простом текстовом редакторе, работающем под управлением MS DOS. Однако наиболее удобно вести разработку ASM файла в специальном текстовом редакторе `avredit.exe`.

На рис. 3.13 показано рабочее окно программы текстового редактора, в котором открыт рассматриваемый нами файл `UA4NM.asm`.

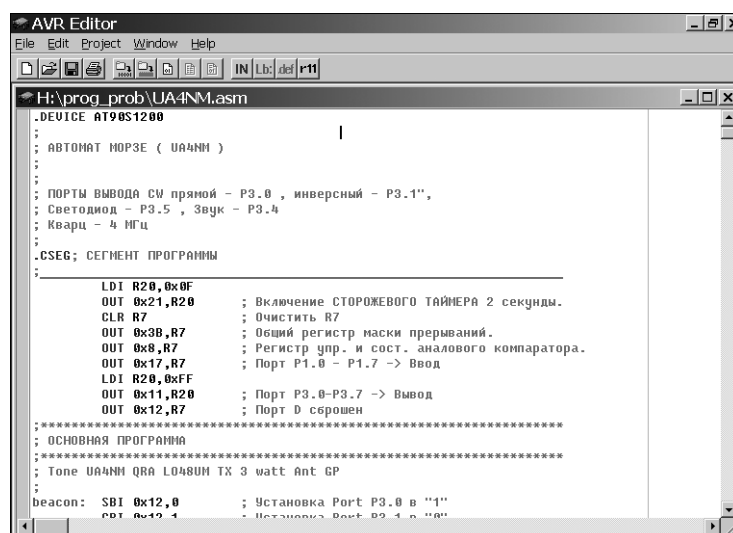


Рис. 3.13. Рабочее окно текстового редактора

Редактор может работать под управлением операционных систем Microsoft Windows 95/98/NT4. Для выделения в нем текста различного назначения используются разные цвета шрифтов. Для ассемблирования редактор использует программу Ассемблера `avrasm.exe`, находящуюся в этой же директории, что и сам редактор. Ошибки ассемблирования выводятся непосредственно в окне редактора.

Подготовленный программой ассемблера файл с расширением `HEX` может быть загружен в память микроконтроллера с помощью программатора.

В листинге 3.5 показан текст файла `UA4NM.hex`.

**Листинг 3.5.** Файл `UA4NM.hex`

---

```
:100000004FE041BD77247BBE78B877BA4FEF41BB54
:1000100072BA909A9198959A40E7D0D04A95E9F7AC
:10002000919A90989598C7D0C1D098D097D09FD0EA
:10003000BBD094D09CD0B8D091D090D08FD08ED05F
:1000400096D0B2D094D08AD0AFD091D090D0AED04C
:10005000B2D08DD08CD082D08AD0A6D088D0A4D077
:100060007DD07CD07BD07AD0A1D078D077D09CD0F6
:1000700075D074D073D09AD07AD070D078D094D014
:100080006DD06CD091D06AD072D068D08DD06FD046
:100090006ED06DD089D062D061D060D068D08BD066
:1000A00066D065D05BD063D07FD058D060D056D0BA
:1000B0007BD054D05CD07AD051D050D075D04ED0B7
:1000C0004DD04CD073D04AD052D048D047D06CD00D
:1000D0004ED04DD04CD068D041D040D03FD03ED053
:1000E00046D062D044D043D042D038D037D05CD054
:1000F00035D034D03CD058D03AD039D038D05BD07D
:1001000036D052D034D02AD029D031D04FD026D0BA
:1001100025D02DD02CD02BD049D020D028D027D0FE
:1001200043D01CD024D040D022D03ED020D03CD0D0
:1001300015D014D013D03FD011D019D035D017D04E
:100140000DD032D014D032D012D011D007D02CD054
:1001500005D00DD00CD002D02ED05BCF909A9198C4
:10016000959A2CD0919A909895980ED00895909A3F
:100170009198959A23D022D021D037E220D0919A1D
:100180009098959801D0089530E85FEFA895A895CC
:10019000A8955A95D9F75FEFA895A895A8955A956F
:1001A000D9F73A9591F708953FEFEFCF3FEFEDDFA5
:1001B0003FEFEBDF0895FADFF9DF089530E75FEFF7
:1001C000A895949A5A95E1F75FEFA89594985A9557
:0801D000E1F73A95A1F708954B
:00000001FF
```

---

Каждая строка начинается с двоеточия — это признак начала строки. Следующие 8 символов представляют собой адрес начала участка памяти, в которую должны быть записаны все остальные символы в этой строке. И так для каждой строки.

---

Сейчас мне вспомнился период середины 1980-х годов, когда вручную подобным образом с клавиатуры компьютера вводились в память тысячи символов! Таким путем в память компьютера вводился текст какой-то программы, затем эта программа из памяти копировалась на магнитофонную ленту и уже в таком виде могла храниться долгое время и распространяться среди друзей.

---

Еще более удобным инструментом для создания и работы с файлами исходных текстов типа ASM является программа AVR Studio, в которой имеются и текстовый редактор, и Ассемблер, и отладчик, и программатор. Но самым важным свойством этой среды разработки программ является ее способность выполнять отладку разрабатываемой программы без подключенного микроконтроллера.

На рис. 3.14 показана справочная заставка этой среды разработки, появляющаяся при выборе пункта меню **About**.

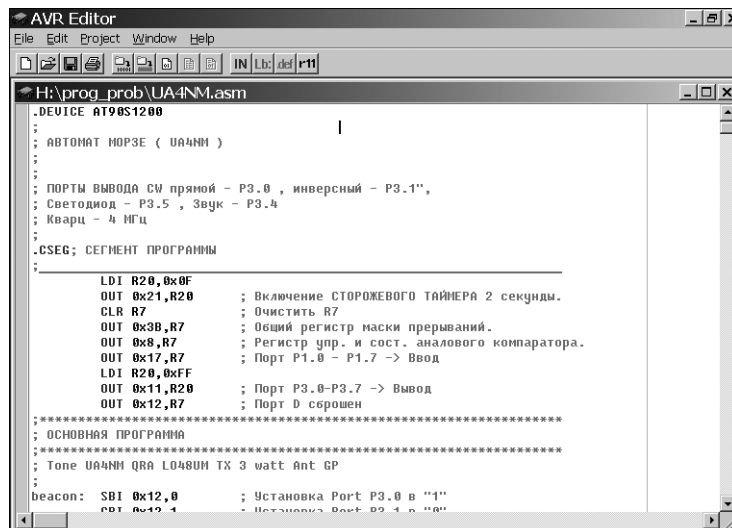


Рис. 3.14. Справочное окно **About**

На рис. 3.15 показано рабочее окно AVR Studio с размещенным в нем текстом файла UA4NM.asm.

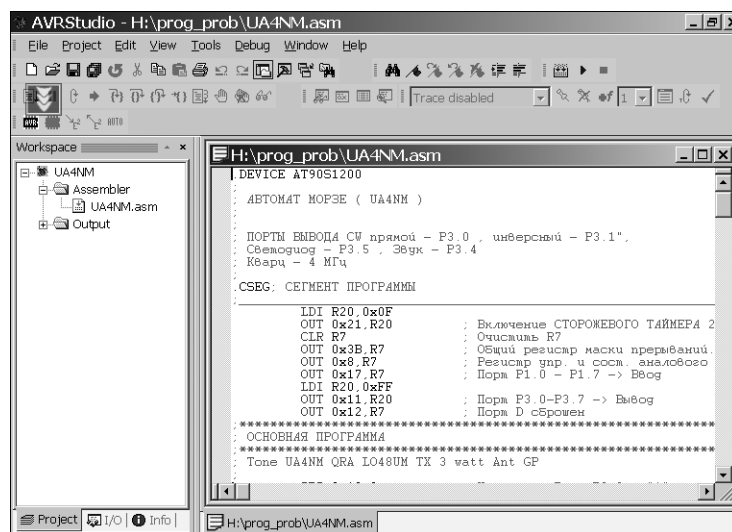


Рис. 3.15. Рабочее окно AVR Studio

Если теперь выбрать пункты меню **Assembler** и **Build**, то в той же директории, в которой находится исходный файл `UA4NM.asm`, программой автоматически будут созданы все возможные рабочие файлы, которые могут пригодиться и для отладки, и для размещения в памяти микроконтроллера. На рис. 3.16 как раз и показаны названия всех созданных файлов.

Если компьютер соединен посредством соответствующего адаптера с микроконтроллером, то AVR Studio может выполнить заключительный этап программирования — ввести необходимые коды в память микроконтроллера.

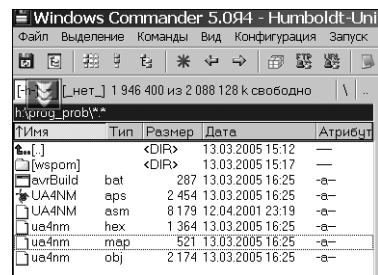


Рис. 3.16. Перечень созданных программой AVR Studio файлов

## Другие программаторы

На различных сайтах в Internet можно найти большое количество самых разнообразных программ, предназначенных для ввода в память микроконтроллера (или иной микросхемы памяти) необходимых исходных данных из соответствующего файла. Некоторые из них можно найти на прилагаемом к данной книге компакт-диске. Но среди этого многообразия мне хотелось бы выделить универсальный программатор **PonYProg200**, который очень популярен среди радиолюбителей тем, что может работать практически со всеми встречающимися и доступными нам микроконтроллерами и микросхемами памяти. На рис. 3.17 показано рабочее окно русифицированного варианта **PonyProg2000**.

## Быстрая разработка программ

Хочу рассказать вам еще об удивительном программном продукте — это среда быстрой разработки программ для микроконтроллеров **AVR Algorithm Builder v440**. Разработку представляет Геннадий Громов (г. Тула), адрес в Internet [home.tula.net/algrom/russian.html](http://home.tula.net/algrom/russian.html).

Данная среда обеспечивает полный цикл разработки, начиная от ввода алгоритма, включая отладку, и заканчивая внутрисхемным программированием кристалла. Вы будете иметь возможность разрабатывать программы как на уровне ассемблера, так и на макро-уровне, при котором возможна работа со знакопеременными величинами произвольной длины. Это приближает возможности программирования к языку высокого уровня.

Графические технологии создания программы раскрывают новые возможности для программистов. Они позволяют вводить исходные коды программы на плоскости рабочего окна в виде алгоритма с древовидной структурой. В результате вся логическая структура программы становится полностью наглядной. Основным предназначением таких технологий является максимальное приведение интерфейса разработки к природе человеческого восприятия. Освоение такой среды намного проще, чем освоение классического ассемблера. Более удобный интерфейс раскрывает новые возможности для разработки. По оценке пользователей, время создания программного обеспечения сокращается в 3–5 раз по сравнению с классическим ассемблером.

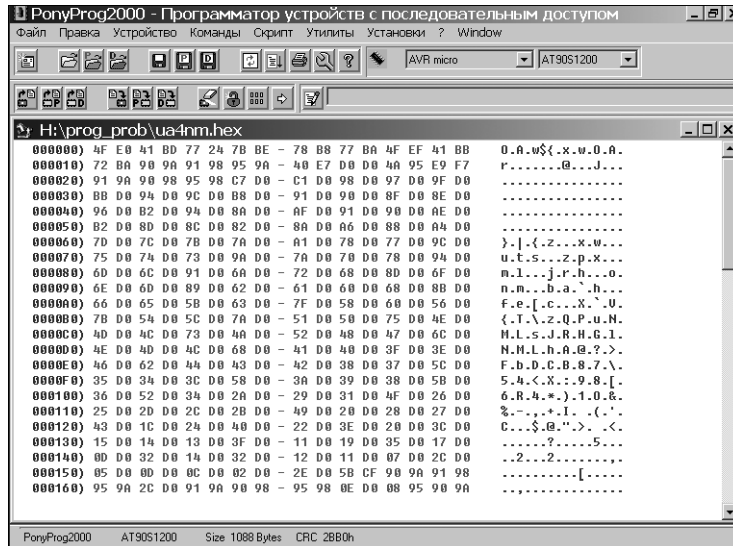


Рис. 3.17. Рабочее окно PonyProg2000

Среда предназначена для работы под управлением операционных систем Microsoft Windows 95/98/ NT/2000/ME/XP. Для нормальной работы редактора в систему должен быть установлен шрифт «Courier».

Любое программное обеспечение можно разбить на отдельные логически завершенные фрагменты. Как правило, финальным оператором этих фрагментов являются такие операторы как безусловный переход или возврат из подпрограммы, т.е. операторы, после которых линейное исполнение программы однозначно прекращается.

Разработка программного обеспечения в среде Algorithm Builder сводится к формированию таких блоков, размещению их на плоскости и установлению между ними векторных связей из условных и безусловных переходов.

На рис. 3.18 показано рабочее окно программы Algorithm Builder с изображенными графическими картинками алгоритмов.

Важной особенностью этой разработки является четко изложенная на русском языке подробная документация и по работе с программой, и по конструированию схем алгоритмов. На мой взгляд, это очень перспективная разработка, которая предоставит программисту, овладевшему этой средой, большое превосходство над другими.

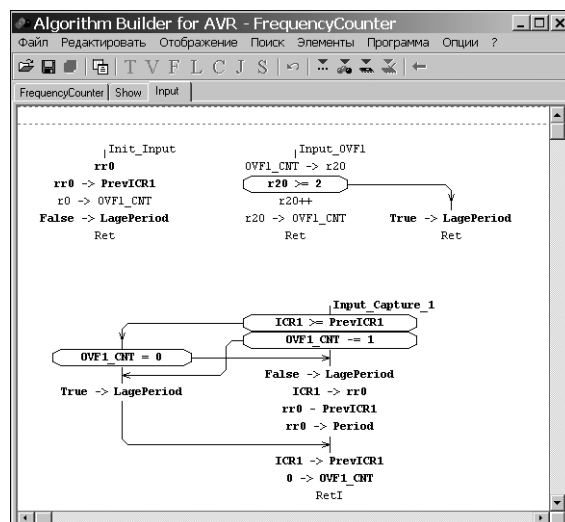


Рис. 3.18. Рабочее окно Algorithm Builder

## Вместо заключения

При работе над материалом этой главы ставилась задача познакомить читателя с устройствами автоматики, которые называются «микроконтроллеры», с методами создания программ, под управлением которых микроконтроллеры работают в автоматическом режиме и как можно более понятно рассказать о том, как эти программы вводятся в память микроконтроллера. Иными словами, познакомить вас с методами программирования микроконтроллеров.

В прилагаемом к книге компакт-диске находится самое разнообразное программное обеспечение, которое только может вам пригодиться при создании своих автоматических устройств на базе микроконтроллеров AVR.

Тем, кто хочет сам научиться программировать на языке Ассемблера, могу посоветовать начать это занятие с подробнейшего рассмотрения и детального разбора чужих программ. При этом начинать нужно с самых простых. Естественно, что для подобных экспериментов потребуется хороший адаптер — это такое аппаратное устройство, которое соединяет компьютер с микроконтроллером. На компакт-диске имеется несколько вариантов разных по сложности адаптеров. Выбирайте любой и программируйте.