Микроконтроллеры? Это же просто!

В процессе общения с авторами и подписчиками журнала мы столкнулись с тем, что многие из них хотели бы освоить работу с микроконтроллерами, но испытывают при этом дефицит литературы, рассчитанной на новичков, самостоятельно начинающих почти с нуля. Посмотрев с этой точки зрения на появившиеся в последние несколько лет посвященные микроконтроллерам книги, а также на публикации в журналах, автор настоящих строк понял, что практически все они направлены на тех, кто уже ориентируется в этой предметной области. Статей и книг, рассчитанных на новичков, и позволяющих им шаг за шагом освоить микроконтроллеры, не перегружая их головы раньше времени важными, но необязательными в первый момент подробностями, увы, нет. Так родилась идея написать цикл статей для начинающих, знакомство с которым позволило бы им осознать, что такое микроконтроллеры, как они устроены, как функционируют, как писать, отлаживать и заносить в них программы, и т. д.

Несмотря на большой опыт написания самых разнообразных статей, подобную задачу автор ставит перед собой впервые. Поэтому тех, кто попытается по подготовленным мной материалам пройти этот путь, я попрошу не стесняясь задавать вопросы, если что-то окажется непонятным или плохо изложенным – мой электронный адрес приведен ниже.

Введение

Название настоящей статьи "слизано" из серии популярных в семидесятые годы книжек, посвященных радио и телевидению, и рассчитанных, также как и настоящий цикл статей, на начинающих. Любой, даже самый сложный вопрос, можно объяснить просто и доступно, и авторам тех книжек это вполне удалось. Постараюсь это сделать и я.

В качестве объекта изучения мы с вами выберем микроконтроллеры (МК) семейства x51. Предполагаю, что некоторые из тех, кто прочел эти строки, уже поморщились – зачем, дескать, писать об этом старье, сейчас есть много гораздо более интересных контроллеров. Да, есть. Но во-первых мне, автору, легче объяснять материал на основе того, что я знаю очень хорошо (AVR или PIC, к примеру, я знаю хуже). Во-вторых, все, кто разобрался хотя бы с одним контроллером, после этого всегда в состоянии самостоятельно разобраться с любым иным – было бы время и желание (или необходимость). В-третьих, с этими контроллерами по-прежнему работает не меньше разработчиков, чем с AVR или PIC-контроллерами, не говоря уже о любых других, а Atmel и Analog Devices в последнее время предоставили в наше с вами распоряжение еще более совершенные образцы контроллеров этого семейства. А в-четвертых, и это главное, те, кто поморщился – это люди, уже разбирающиеся в микроконтроллерах. Господа разбирающиеся! Этот цикл статей не для вас! Здесь вы вряд ли найдете что-то новое для себя. Читать этот материал я вам рекомендую лишь в том случае, если вы замыслите написать подобные статьи по другим типам контроллеров. Тогда читайте то, что написано мной, фиксируйте, что изложено удачно, а что нет, и готовьте свои статьи с учетом возможных огрехов, увы, неизбежных в первой редакции такого материала. Ну а мы с теми, кто еще не разбирается в микроконтроллерах, потихоньку двинем дальше.

Глава 1. Первое знакомство

Память микроконтроллера

Первое, с чего стоит начать – это со слов, что при всей кажущейся сложности ничего непостижимого в микроконтроллерах нет. МК представляют собой микросхемы, которые всего-навсего "от корки до корки" выполняют программы, занесенные в них программистами. Последние, зная, что из себя представляет микроконтроллер, а также то, как и какие команды он может выполнить, составили и отладили программы (последовательность этих самых команд), занесли их в микроконтроллеры, и при подаче питания МК выполняют все то, что было предусмотрено программистами.

Из сказанного выше вытекает ряд вопросов. Вопрос номер один – микроконтроллер может выполнять какие-то команды. Какие? Второй вопрос – программа пишется программистом и заносится в МК. Как? А заодно, где она там хранится?

Начнем с последнего. У большинства МК имеется **память программ**, представляющая из себя некоторое количество ячеек (от тысячи до десятка тысяч и более). Она находится внутри самой микросхемы (говорят – "на кристалле"). Каждая ячейка имеет свой порядковый номер, или, как говорят программисты, адрес. В этих ячейках хранятся числа, могущие принимать значения от 0 до 255. Упомянутые числа и есть та самая программа, которую выполняет микроконтроллер, если мы подадим на него питание.

Удивлены? На самом деле все довольно просто, хотя на первый взгляд может показаться непривычным. Программа – это последовательность выполняемых микроконтроллером команд. Каждой команде в памяти программ соответствует свое число (корректнее сказать – код). При включении питания МК один за другим считывает эти коды, осуществляет их дешифрацию (другими словами, определяет, что же нужно сделать), а затем исполняет одну за другой эти дешифрированные команды. Кстати, отмечу, что в контроллерах семейства x51 первой выполняется команда, код которой расположен в самой первой ячейке памяти программ с адресом 0.

Главная особенность памяти программ – занесенные в нее коды сохраняются неизменными при отсутствии питания у микроконтроллера. Это, в общем, и очевидно – уж коль скоро мы написали и отладили программу, она не должна самопроизвольно изменяться с течением времени.

Память программ – это не единственный вид памяти, имеющийся внутри микроконтроллера. Любой МК имеет еще **память данных**. Ее принципиальное отличие от памяти программ состоит в том, что микроконтроллер может не только читать содержимое ее ячеек, но и определенными своими командами содержимое их изменять (записывать в них данные), в то время как менять содержимое памяти программ ему "не по зубам". В силу описанной причины память данных еще иногда называют оперативной памятью (оперативным запоминающим устройством, или ОЗУ), в отличие от памяти программ, именуемой постоянным запоминающим устройством (ПЗУ).

Кроме того, стоит отметить, что занесенные в ОЗУ коды теряются (т. е. изменяются произвольным образом) при выключении питания.

Последний находящийся внутри МК вид памяти, без знакомства с которым мы не сможем двигаться дальше – это так называемая **регистровая память**, или регистры. Они представляют из себя ячейки оперативной памяти, обращение к которым контроллер осуществляет более короткими и быстровыполнимыми командами, чем к упомянутому в предыдущем абзаце ОЗУ. В остальном же регистры и ячейки оперативной памяти идентичны – содержимое их теряется при выключении питания, и может быть считано или изменено при выполнении микроконтроллером некоторых из своих команд. Более подробно о регистрах (их названиях, адресах, командах обращения

к ним) мы будем говорить в одной из следующих глав.

Для пояснения всего сказанного приведу следующий пример. На рис. 1 показано содержимое последовательно расположенных восьми ячеек памяти программ микроконтроллера с занесенным в него куском пользовательской программы. На первый взгляд эта сущая бессмыслица. Но на самом деле это – вполне разумный фрагмент реальной программы. Код 238 предписывает микроконтроллеру перенести (говорят "прочитать") число из регистра R6 в главный регистр МК, называемый аккумулятором. Код 36 предписывает контроллеру прибавить к содержимому аккумулятора число, размещенное в памяти программ непосредственно за этим кодом (т. е. в данном случае – единицу). Код 254 предписывает вернуть полученное в результате суммирования число из регистра-аккумулятора в регистр R6. Следующий код (239) вынуждает МК прочитать в аккумулятор число из регистра R7. Код 52 – прибавить к содержимому аккумулятора число, расположенное в программной памяти за этим кодом (т. е. 0), и еще так называемый бит переноса (для любопытных – бит переноса равен 1, если предыдущее суммирование дало результат больше или равный 256, и 0, если меньше; вспомните единичку, которая "идет на ум" при суммировании на бумажке "столбиком", когда сумма оказывается больше 9. Бит переноса – это и есть та самая единичка). Последний код – 255 – предписывает вернуть число из аккумулятора в регистр R7.

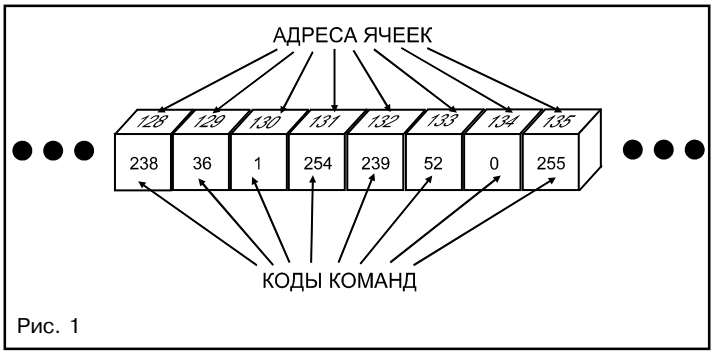

Рис. 1

Не будем обсуждать, зачем нужно выполнять указанную последовательность команд – поверьте на слово, иногда нужно. Смысл приведенного фрагмента – показать, что между числами в памяти программ и конкретными командами есть взаимно однозначное соответствие. Программы, которые пишутся пользователями, переводятся в коды при помощи важной и нужной программы, именуемой транслятором с языка ассемблера, или просто ассемблером (далее мы с ним обязательно познакомимся). После этого полученные коды заносятся в память программ микроконтроллера. Для этой цели служат специально выпускаемые рядом фирм устройства, именуемые программаторами. Микроконтроллер вставляется в панельку программатора (только предварительно нужно убедиться, что используемый программатор рассчитан на программирование вашего контроллера), запускается соответствующая программа, в которой указывается тип контроллера и имя файла, в котором находятся коды вашей программы, и программатор заносит их в МК. Если, конечно, последний исправен.

Кстати, существуют программы, называемые дизассемблерами, которые осуществляют обратную операцию – переводят коды программы в понятные человеку команды.

Особенности включения МК и назначение выводов

Оставим теперь на время внутреннее устройство МК и обратимся к внешнему. Стандартные микроконтроллеры семейства x51 выпускаются в сороконогих DIP-корпусах с расстоянием между рядами ножек 15 мм, а между самими ножками – 2,54 мм. Их цоколевка и стандартная схема включения приведена на рис. 2.

Ножка 20 – GND (земля). Она соединяется с общим проводом. Ножка 40 (Vcc) – с шиной питания (+3...5 В). К ножкам 18 (XTAL2) и 19 (XTAL1) подключается кварцевый резонатор. Наиболее часто используемые кварцы – на 11,0592 МГц и 12 МГц, хотя на практике МК семейства x51 работают с кварцами и с более низкими частотами (например, 1 Мгц), и с более высокими (я встречал МК x51 от Philips, работающий на 40 МГц). Для более стабильного запуска выводы кварцевого резонатора соединены с общим проводом через конденсаторы C1 и C2 емкостью от 15 до 30 пФ.

Кстати, в англоязычной литературе ножки микросхем называются пинами (pin – булавка, шпилька, гвоздь, болт, штифт).

Ножка 9 – это вход RESET, или СБРОС. Единичный уровень на этом входе в течение нескольких десятков периодов тактового генератора приводит к сбросу в начальное состояние регистров микропроцессора и к началу исполнения программы с нулевого адреса. Сброс обязателен при подаче напряжения питания на микроконтроллер. С этой целью вход RESET соединяют с шиной питания через конденсатор C3 емкостью несколько микрофарад и с общим проводом – через резистор R1 сопротивлением порядка сотни килоом. В момент включения питания конденсатор разряжен, и вход сброса оказывается под потенциалом, близким к Vcc. Несмотря на снижение этого потенциала вследствие заряда C3 в течение нескольких десятков миллисекунд уровень сигнала на входе сброса остается единичным, и осуществляется корректный запуск микроконтроллера.

Как-то раз в одной из моих плат вследствие ошибки в разводке вход сброса оказался висящим в воздухе – упомянутая RC-цепь была соединена с соседней ножкой. В результате мне понадобилось несколько дней для того, чтобы понять, почему при включении питания контроллер то запускался нормально, а то "зависал", не подавая никаких признаков жизни. Я запрограммировал второй контроллер, вставил вместо первого – то

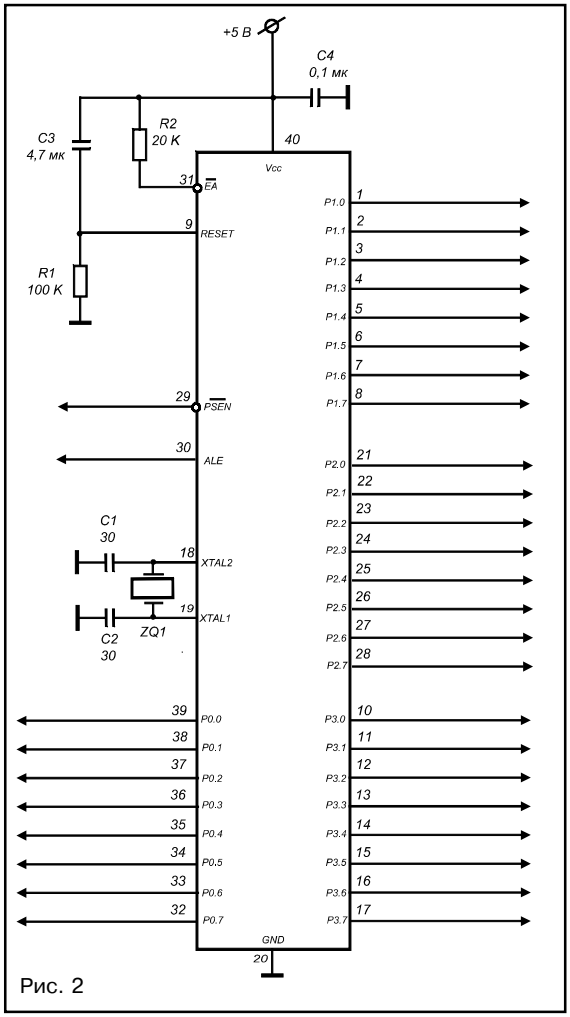

Рис. 2

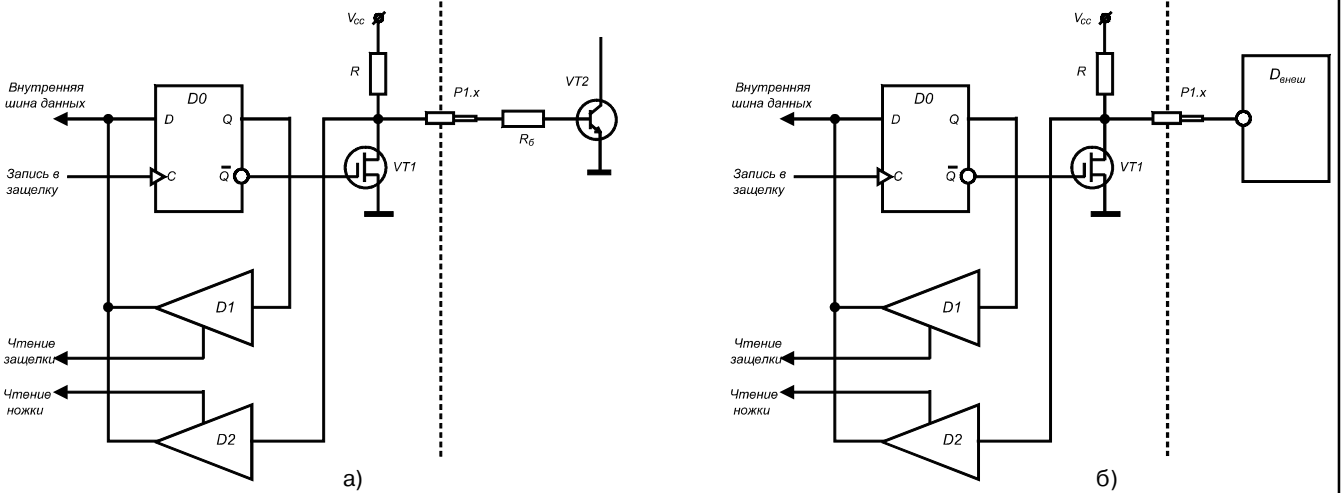

Рис. 3

же самое, разве что после этого один нормальный старт стал приходится не на четыре "зависания", а на три. В общем, если микроконтроллер с отлаженной программой то нормально стартует, а то ведет себя кое-как, начните с проверки цепи сброса.

Следующий важный вход – EA, ножка 31. Если на него подана логическая единица, то МК работает с уже упоминавшейся памятью программ, расположенной на кристалле. Нуль на входе EA заставит микроконтроллер выполнять программу из внешней памяти (такое возможно). О том, как организовывается связь между МК и дополнительной микросхемой, содержащей эту внешнюю память, мы расскажем в одной из следующих глав. На первых же порах мы будем работать только с памятью программ на кристалле, поэтому на входе EA должна быть установлена логическая единица. Избегайте плавающего потенциала на этом входе – если он окажется висящим в воздухе, контроллер будет работать нестабильно, постоянно сбоить и "зависать".

На ножке 30 (ALE) обычно присутствует непрерывная последовательность прямоугольных импульсов с частотой в 6 раз ниже, чем у кварцевого резонатора, соединенного с ножками 18 и 19. Для 12-мегагерцового кварца она, очевидно, составит 2 МГц. В этой последовательности длительность единицы на ножке ALE примерно вдвое ниже длительности нуля, т. е. скважность составляет 33%. Этот сигнал можно использовать для тактирования микросхем, требующих для работы внешний источник тактового сигнала.

Назначение ножки 29 (PSEN) будет рассмотрено в разделе, где мы будем говорить о подключении к МК внешней памяти.

Оставшиеся 32 ножки – это линии ввода/вывода информации. Они сгруппированы по восемь в четыре так называемых порта ввода/вывода (P0, P1, P2 и P3). Каждая линия любого из них может использоваться либо как вход, либо как выход, независимо от использования остальных линий. Для этого их оконечные каскады выполнены соответствующим образом. На рис. 3 приведена упрощенная схема одной из линий ввода/вывода порта P1.

Как видно из рисунка 3а, ножка микросхемы P1.x соединена со стоком выходного полевого транзистора VT1, "подтянутого" к потенциалу питания при помощи внутреннего нагрузочного резистора R. Одновременно с этой же ножкой микросхемы соединен вход буфера ввода D2. Если мы присоединим к этой ножке микросхемы через резистор $R_б$ базу внешнего транзистора VT2, то занося в триггер-защелку D0 логические 1 или 0, мы будем открывать или закрывать VT2, реализуя таким образом выбранную линию в качестве линии вывода информации.

Использование линии в качестве линии ввода информации иллюстрируется рис. 3б. Ножка микроконтроллера (а, следовательно, и вход буфера D2) соединены с выходом микросхемы $D_{внеш}$, состояние которого мы хотим проанализировать (иными словами, "ввести" его в МК или "прочитать"). Но прежде, чем читать содержимое буфера D2, необходимо закрыть транзистор VT1, записав в триггер D0 этой линии единицу. В самом деле, если VT1 будет открыт, он попросту шунтирует анализируемый выход микросхемы $D_{внеш}$. В лучшем случае этот конфликт на ножке просто исказит вводимую информацию, когда выход микросхемы $D_{внеш}$ будет "тянуть" потенциал вверх, а выход VT1 – вниз. В худшем же варианте победа сильнейшей из противоборствующих сторон приведет к сгоранию слабейшей. Так что запомним, что если какие-то линии порта мы собираемся использовать в качестве линий ввода, то перед этим обязательно в соответствующие выходные триггеры нужно записать единицы.

По схеме, приведенной на рис. 3, выполнены линии портов P1, P2 и P3. Порт P0 оформлен несколько иначе – сток его транзистора VT1 вместо обычного нагрузочного резистора соединен с динамической нагрузкой (источником тока). Это сделано для того, чтобы линии порта P0 при занесении в их триггеры-защелки единичек оказывались в так называемом высокоимпедансном ("сером") состоянии, характеризующимся очень высоким выходным сопротивлением. В остальном же функционирование линий порта P0 похоже на работу линий остальных трех портов.

Александр Фрунзе,
alex.fru@mtu-net.ru

Продолжение следует

Литература

1. *А. Фрунзе, С. Хоркин. Однокристальные микро-ЭВМ. РАДИО, 1994 г., №№ 8–12/1994, №№ 1–3/1995.*

2. *В. Я. Нерода, В. Э. Торбинский, Е. Л. Шлыков. Однокристальные микро-ЭВМ MCS-51. Архитектура. – М., изд. Диджитал Компонентс. 1995, с. 164.*

3. *Р. Токхайм. Микропроцессоры: курс и упражнения. Пер. с англ. – М., Энергоатомиздат. 1988, с. 336.*

4. *В. В. Сташин, А. В. Урусов, О. П. Мологонцева. Проектирование цифровых устройств на однокристальных микроконтроллерах. – М., Энергоатомиздат, 1990, с. 224.*