Оглавление

Вступительная статья 5
Предисловие к второму изданию 7
Предисловие 9

Глава 1. **Первое знакомство** 12

1.1. Память микроконтроллера 12
1.2. Особенности включения микроконтроллера и назначение выводов 15
1.3. Двоичные и шестнадцатеричные числа 19
1.4. Краткие выводы 22

Глава 2. **Сопряжение МК с программно-управляемыми ИС** 24

2.1. Сопряжение с параллельным АЦП 24
2.2. Программа работы с параллельным АЦП 28
2.3. Ассемблер: основные понятия и приемы 34
2.4. Ассемблер: особенности трансляции 40
2.5. Ассемблер: ошибки трансляции 47
2.6. Сопряжение с последовательным АЦП 50
2.7. Краткие выводы 57

Глава 3. **Регистры микроконтроллера** 60

3.1. Регистры общего назначения и слово состояния программы 61
3.2. Аккумулятор, расширитель аккумулятора, указатель стека и подпрограммы 65
3.3. Работа МК с внешней памятью данных 70
3.4. Регистр-указатель данных 79
3.5. Пример: подпрограмма, использующая регистры МК 80
3.6. Краткие выводы 92

Глава 4. **Сопряжение МК с индикаторами различных типов** 94

4.1. Сопряжение с ЖКИ на основе контроллера HT1611 фирмы HOLTEK 95
4.2. Сопряжение со светодиодными индикаторами типа АЛС318 104

4.3. Сопряжение с ЖКИ на основе контроллера HD44780 фирмы HITACHI 115
4.4. Сопряжение с 4-разрядными светодиодными матричными индикаторами 136
4.5. Краткие выводы 163

Глава 5. **Система команд микроконтроллеров x51** 166

5.1. Общие сведения о системе команд 168
5.2. Группа команд передачи данных 172
5.3. Группа команд арифметических операций 180
5.4. Группа команд логических операций 185
5.5. Группа команд операций с битами 187
5.6. Флаги результата 190
5.7. Группа команд передачи управления 191
5.8. Краткие выводы 200

Глава 6. **Таймеры-счетчики и система прерываний МК x51** 202

6.1. Таймеры-счетчики микроконтроллеров семейства x51 202
6.2. Система прерываний микроконтроллеров семейства x51 209
6.3. Программа, использующая таймер-счетчик и прерывание 216
6.4. Краткие выводы 222

Глава 7. **Практические примеры разработки устройств на МК x51** 225

7.1. Противоугонное устройство на микроконтроллере семейства x51 225
7.2. Простой термостабилизатор на микроконтроллере AT89C2051 234
7.3. Подпрограммы целочисленного многобайтного умножения и деления 240
7.4. Милливольтметр постоянного тока на АЦП AD7894 и МК семейства x51 253
7.5. Милливольтметр постоянного тока на АЦП AD7714 и МК семейства x51 264
7.6. Краткие выводы 284

Глава 8. **Использование приемопередатчика для связи с ПК** 286

8.1. Как связывать микроконтроллер и компьютер по каналу RS-232 287
8.2. Описание микросхем 289
8.3. Режим работы микроконтроллера с последовательным каналом 290
8.4. Основные подпрограммы для микроконтроллера 290
8.5. Общая программа для микроконтроллера. Диаграмма состояний устройства ... 293
8.6. Общая программа для ПК. Диаграмма состояний ПК 299
8.7. Основные подпрограммы для ПК 301
8.8. Пользовательское описание программы для ПК 305
8.9. Заключение 308

Литература 310

Вступительная статья

Как известно, уже набила оскомину фраза про «все возрастающий интерес инженерной общественности к применению микроконтроллеров в задачах управления и обработки сигналов». Литературы по упомянутой тематике вроде бы стало издаваться достаточно. Тем не менее, выход данной книги — событие. Почему же? Так потому, что — ПРОСТО! Лично меня трудно удивить литературной новинкой в электронной области, но когда я взял в руки первую статью из цикла, на базе которого написана эта книга, то испытал приятный шок — наконец-то на просторах 1/6 части суши появилась книга, написанная на хорошем русском языке, не лишенная чувства юмора и вместе с тем технически исключительно цельная и грамотная. В общем, это и неудивительно. Александр Виленович Фрунзе — известный и авторитетный специалист, и хочется надеяться, что книгу, которую вы сейчас держите в руках, ждет долгая и счастливая судьба. Я почти уверен, что и через 10...15 лет основные ее положения будут оставаться актуальными. Это важное отличие данной книги от великого множества изданий по микроконтроллерной тематике, которые в огромном большинстве представляют собой либо справочники, либо компиляцию с фирменных руководств по конкретным семействам.

Материал излагается на примере старого доброго 51-го контроллера. Предвижу возмущение некоторых специалистов, готовых похоронить этот контроллер из-за «неперспективности и устаревшей архитектуры и системы команд». Тем не менее, на основе этой архитектуры выпускается огромное число различных контроллеров, содержащих на борту широкую гамму переферийных устройств — от портов до АЦП, ЦАП и интерфейса CAN. Кроме того — как известно, «кадры решают все». В стране выросло не одно поколение разработчиков, активно применяющих 51 контроллеры, его изучение введено в учебные планы вузов, наконец в классическом исполнении его еще выпускает отечественная промышленность. Из личного опыта могу сказать только одно — используй ту элементную базу, которой владеешь. К сожалению, внедрение новых семейств связано с необходимостью их освоения — а это время и, конечно же, деньги.

Материал книги изложен методически исключительно грамотно. Она самодостаточна — начинающему не придется безуспешно искать в библиотеках издания, на которые обычно ссылаются при рассмотрении смежных вопросов (как то машинная арифметика, представление чисел и т. п.), в то же время опытный разработчик легко может пропустить знакомые разделы без ущерба для понимания. Изложение настолько последовательно и выверено, что книга читается как хорошее художественное произведение. Крайне разумно используются приложения — они позволяют не перегрузить основной текст и в то же время это не сухие справочные данные, а разделы, имеющие самостоятельную ценность. Тут и описание работы с микроконверторами от Analog Devices, и исследование влияния на рынок средств поддержки разработки, и пути модернизации систем — охват тем широчайший! Причем и в основном тексте, и в приложениях чувствуется авторский стиль.

Отдельно следует упомянуть о примерах из книги. Не поленился и прогнал первый попавшийся наугад — РАБОТАЕТ! Все примеры тщательно протестированы и помогут начинающему в освоении нелегкого искусства разработки систем на микроконтроллерах.

Конечно, как и в любой столь масштабной работе, не обошлось без недостатков. Практически не рассматриваются вопросы проектирования систем с несколькими параллельно работающими контроллерами.

Особо следует сказать, что эту книгу можно и нужно рекомендовать студентам. Она стимулирует самостоятельность мышления и дает представление о логике и методах проектирования, что зачастую гораздо важнее технических подробностей.

Доцент кафедры
«Автономные информационные
и управляющие системы»
МГТУ им. Н.Э.Баумана,
к.т.н. В.Б.СТЕШЕНКО

Предисловие к второму изданию

Переиздание этой книги — событие для меня, безусловно, приятное. Я не сомневался, что книга будет переиздаваться, но одно дело — мои соображения по тому или иному вопросу, а другое — события, их подтверждающие.

Второе издание практически полностью повторяет первое. Я лишь исправил несколько досадных «очепяток», имевших место преимущественно в шестой главе, да добавил абзац во второй главе в то место, где описывается, как с помощью моего ассемблера получить hex-файл (этот момент в первом издании был сформулирован нечетко, и некоторые читательские письма содержали просьбы о дополнительном разъяснении). Все остальное в книге оставлено без изменений.

С момента выхода книги я получил от ее читателей более двух сотен писем. Самыми ценными для меня являются те пять, в которых авторы просто благодарят меня за книгу, без приложения к благодарности того или иного вопроса. Ведь их писать заставила не нужда получить от меня дополнительно ту или иную подсказку, а именно благодарность.

Но и тем письмам, которые вместе с благодарностью содержали какие-либо вопросы, я был рад. Анализ этих вопросов показал, что и где мною было сформулировано недостаточно точно, и куда вкрались опечатки. Так что и эти письма тоже были важны — именно они позволили исправить упомянутые недостатки.

Помимо конкретных технических вопросов, авторы писем спрашивали и о том, почему в этом томе столь мало внимания уделено периферийным устройствам, интерфейсам и т. д. Ответ, в общем-то, прост. Главное назначение этой книги — помочь тем, кто только начинает осваивать микроконтроллерную технику. А для них подробное описание I^2C-интерфейса или аналоговой периферии новейших LPC-микроконтроллеров — это китайская грамота, излишняя на первом этапе знакомства. Формируя книгу, я старался оставить в ней лишь то, без чего научиться работать с микроконтроллерами действительно невозможно. И, судя по отзывам читателей, мне это

вполне удалось. Именно поэтому она оставлена практически без изменений.

Когда готовилось к выпуску первое издание первых двух томов книги, я еще не предполагал, что у нее будет продолжение. Однако безусловный читательский интерес к ней стимулировал работу над третьим и четвертым томами. Материалы, помещенные в этих томах, очень полезны не только для формирования правильного стиля программирования, но и для расширения кругозора разработчика.

А. В. Фрунзе

Предисловие

Идея написания этой книги возникла у меня довольно неожиданно. Так сложилось, что в течение одной-двух недель сразу четверо авторов из тех, кто регулярно публикует свои материалы в журнале «СХЕМОТЕХНИКА», в разговоре со мной коснулись темы полного отсутствия литературы, по которой люди, не имеющие опыта работы с микроконтроллерами, смогли бы освоить их. Примерно в это же время я услышал подобные сетования и от двух моих знакомых инженеров-электронщиков, специалистов в аналоговой электронике — они тоже столкнулись с тем, что хотели бы освоить работу с микроконтроллерами, но не представляют, где найти литературу, рассчитанную на новичков, самостоятельно начинающих почти с нуля. Мне казалось, что подобной литературы если не навалом, то, во всяком случае, очень много, и просто нужно пару раз съездить в книжные магазины, торгующие научно-технической литературой — от обилия предлагаемых ими книг просто рябит в глазах, и не может быть, чтобы выбрать было не из чего. Но когда я сам посмотрел на появившиеся в последние несколько лет книги по микроконтроллерной тематике, а также на публикации в журналах, я понял, что практически все они ориентированы на тех, кто уже освоил эту предметную область. Статей и книг, рассчитанных на новичков, и позволяющих шаг за шагом освоить микроконтроллеры, не перегружая, раньше времени важными, но необязательными в первый момент подробностями, увы, нет. Так родилась идея написать для начинающих цикл статей по микроконтроллерам, 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. Тем более что в свое время подобные книги были — достаточно вспомнить, например, замечательную книгу Дж. Коффрона «Технические средства микропроцессорных систем», вышедшую в 1983 году в издательстве «Мир», — с ней знакомы практически все отечественные специалисты по микроконтроллерной технике, начинавшие в 80-х с незабвенного КР580ИК80...

Первоначально я планировал просто написать цикл статей для журнала «СХЕМОТЕХНИКА», полагая, что вряд ли стоит делать книгу из материала, в основе которого лежат тысячу раз описанные микроконтроллеры разработанного еще в конце 80-х годов прошлого столетия семейства x51. Однако несколько десятков отзывов, которые я получил от читателей в ходе публикации первых частей подготовленного материала, и отклики авторов, имеющих опыт издания книг, убедили меня в том, что подобная книга может быть интересна широкому кругу читателей. Поэтому после публикации в «СХЕМОТЕХНИКЕ» первых трех глав я решил, что этот материал должен появиться в виде книги, и готовил его далее уже с учетом принятого решения.

Название настоящей книги «слизано» из серии популярных в семидесятые годы книжек, посвященных радио и телевидению, и рассчитанных, также как и настоящая, на начинающих. Любой, даже самый сложный вопрос, можно объяснить просто и доступно, и авторам тех книжек это вполне удалось. Постарался это сделать и я.

Несмотря на большой опыт написания самых разнообразных статей, задачу подготовки книги для начинающих я поставил перед собой впервые. Тем более что книг аналогичного содержания, как я уже отмечал, у нас пока еще нет. Поэтому мне было довольно сложно принимать решения, какой материал может быть опущен при первом знакомстве с микроконтроллерами, а без какого не обойтись. Кроме того, по собственному опыту я знаю, что если процесс освоения новых знаний идет успешно, то предлагаемого материала никогда не бывает достаточно, и начинаешь искать дополнительную литературу. Поскольку вследствие необходимости упрощения материала основной части книги многие особенности рассматриваемых микроконтроллеров мне пришлось опустить, я счел необходимым снабдить книгу большим количеством приложений, куда и перенес опущенное. Эти приложения выпущены отдельно (Фрунзе А. В. Микроконтроллеры? Это же просто! Т. 2. — М.: ООО «ИД СКИМЭН», 2002). Знакомство с приложениями не является необходимым при первом чтении книги, но, безусловно, окажется полезным для тех, кто решит идти дальше. При этом знакомиться с материалами приложений можно в любой последовательности, по мере понимания приведенной там информации и появления интереса к ней. Обратите также внимание и на список рекомендуемой литературы — некоторые вопросы в этих книгах освещены гораздо полнее, чем у меня, хотя в силу особенности стиля, которым они написаны, начинающим я рекомендовал бы знакомиться с этими книгами лишь после того, как они разберутся с существом рассматриваемой темы по материалам настоящей книги.

Отдельно хочу сказать о программном обеспечении к настоящей книге. Я в своей работе (пока еще) в основном использую старый DOS'овский ас-

семблер TASM. В связи с этим все, что говорится в главе 2 о процедуре ассемблирования, относится именно к этому ассемблеру. Соответственно, фрагменты программ и сами программы также написаны на этом ассемблере. Все это (TASM и программы) вы можете найти на сайте: www.pyrometer.ru .

Для тех, кто уже работает с каким-либо другим ассемблером, при использовании моих программ и подпрограмм может понадобиться некоторое редактирование исходных ассемблерных текстов с учетом особенностей их ассемблеров.

Далее необходимо отметить следующее. Все программы и их фрагменты, приведенные в главах 1—8 книги, взяты из реальных, написанных мной в разные годы программ. Однако при переносе их в книгу и некотором «причесывании» могли возникнуть ошибки. Я тщательно проверял полученные тексты, но 100-процентно гарантировать то, что в приведенных материалах ошибок нет, я все же не рискнул бы. Поэтому если вы обнаружите в моих программах какую-либо ошибку, я очень прошу сообщить об этом мне (alex.fru@mtu-net.ru). Информация об ошибках, если таковые найдутся, будет оперативно отражаться в материалах, помещенных на вышеупомянутых сайтах.

Ряд материалов, вошедших в книгу, подготовлен не мной, а другими авторами. Приложения 5 и 9 написаны Ю. Зобниным и Ш. Кобахидзе, приложение 6 — В. Мясниковым (все они — специалисты московской фирмы «Фитон»). Материал главы 8 и приложения 7 подготовлен моим сыном, Алексеем Фрунзе. Обзор микроконтроллерных семейств фирмы CYGNAL (первая часть приложения 10) подготовлен О. Николайчуком (АО InformInstrument, Кишинев, Молдова), информация по ADuC812 и ADuC816 (приложение 11) — А. Соловьевым и А. Соболевым (фирма «Аргуссофт», Москва). Все перечисленное я только отредактировал. Эти материалы включены в книгу с устного согласия авторов, и я хочу поблагодарить их за проделанную работу.

И в заключение я хотел бы отметить терпение моей жены Татьяны, с пониманием относившейся к более чем полугодовым моим вечерне-ночным бдениям над рукописью, без чего появление книги, которую вы держите сейчас в руках, вряд ли было бы возможным.

А. В. Фрунзе

Глава 1. **ПЕРВОЕ ЗНАКОМСТВО**

Как уже упоминалось в предисловии, в качестве объекта изучения я выбрал микроконтроллеры *семейства x51.* Почему именно их? Во-первых потому, что мне, автору, легче объяснять материал на основе того, что я знаю лучше всего (AVR или PIC, к примеру, я знаю намного 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). А в-третьих, с этими контроллерами попрежнему работает не меньше разработчиков, чем с AVR или PIC-контроллерами, не говоря уже о любых других, причем SiLabs, Atmel и Analog Devices в последнее время предоставили в наше с вами распоряжение еще более совершенные образцы контроллеров этого семейства. Да и не только они — чтобы убедиться в этом, достаточно заглянуть в приведенные в приложении (том 2) обзоры, посвященные x51-совместимым микроконтроллерам, выпускаемым более чем десятком лидеров мировой микроэлектроники. После знакомства с ними становится очевидно, что слухи о кончине славного 51-го семейства оказались явно преувеличенными, и еще добрый десяток лет эти изделия будут вполне конкурентоспособными в семействе наиболее распространенных 8-битных микроконтроллеров. Так что отбросим снобизм и начнем наше знакомство со старым и добрым семейством x51.

1.1. Память микроконтроллера

Первое, с чего стоит начать — это со слов, что при всей кажущейся сложности ничего непостижимого в микроконтроллерах нет. Микроконтроллеры представляют собой микросхемы, которые всего лишь скрупулезно выполняют программы, записанные в них программистами. Последние, зная, что из себя представляет микроконтроллер, какие команды он может выполнить, составили и отладили программы (последовательность этих самых команд) и записали их в микроконтроллеры, которые при подаче питания выполняют все то, что было предусмотрено программистами.

Наверное, у вас уже возникли вопросы. Например: микроконтроллер может выполнять какие-то команды. Какие? Далее: программа пишется программистом и заносится в микроконтроллер. Как? А заодно, где она там хранится?

Начнем с последнего. У большинства микроконтроллеров 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, значения которых могут изменяться от 0 до 255. Совокупность этих чисел и есть та самая программа, которую выполняет микроконтроллер, когда 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 микроконтроллер 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 микроконтроллера. Действительно — 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 микроконтроллер 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». Память данных еще иногда называют оперативной памятью (оперативным запоминающим устройством или ОЗУ), в отличие от памяти программ, именуемой постоянным запоминающим устройством (ПЗУ).

Записанные в ОЗУ коды теряются (т. е. изменяются произвольным образом) при выключении питания.

Помимо ОЗУ и ПЗУ, микроконтроллер содержит еще один вид встроенной 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

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. 1.1** показано содержимое последовательно расположенных восьми ячеек памяти программ микроконтроллера с занесенным в него фрагментом пользовательской программы.

Рис. 1.1. Пример содержимого последовательно расположенных восьми ячеек памяти программ микроконтроллера

На первый взгляд, это 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, 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 микроконтроллер 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.

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 микроконтроллер. Если, конечно, последний исправен...

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

1.2. Особенности включения микроконтроллера и назначение выводов

Оставим теперь на время внутреннее устройство микроконтроллера и обратимся к его внешним цепям. Стандартные микроконтроллеры семейства x51 выпускаются в 40-выводных DIP-корпусах с расстоянием между рядами выводов 15 мм, а между самими выводами — 2.54 мм. Их цоколевка и стандартная схема включения приведена на **Рис. 1.2**.

Вывод 20 — GND (ЗЕМЛЯ). Он, очевидно, соединяется с общим проводом. Вывод 40 (V_{CC}) соединяется с шиной питания (+3...5 В). К выводам 18 (XTAL2) и 19 (XTAL1) подключается кварцевый резонатор. Наиболее часто используемые кварцы — на 11.0592 МГц и 12 МГц, хотя на практике микроконтроллеры семейства x51 работают с кварцами и с более низкими частотами (например, 1 МГц), и с более высокими (я встречал микроконтроллер 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 выводы микросхем называются пинами (от английского слова *pin* — булавка, шпилька, штифт).

Вывод 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 микроконтроллера и к началу исполнения программы с нулевого адреса. Сброс обязателен при по-

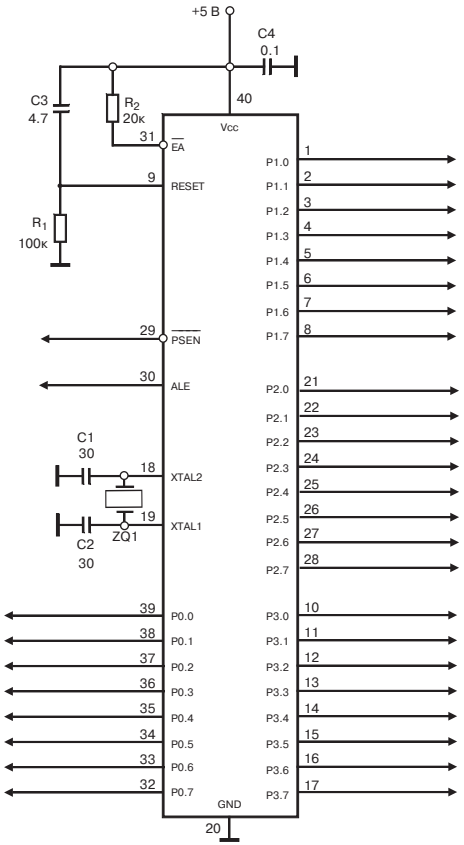


Рис. 1.2. Цоколевка и стандартная схема включения микроконтроллера семейства x51

даче 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 напряжению питания. Несмотря на снижение этого потенциала вследствие заряда C3, в течение нескольких десятков миллисекунд уровень сигнала на входе сбро-

са 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 соседним выводом. В результате мне понадобилось несколько дней для того, чтобы понять, почему при включении питания контроллер то запускался нормально, то «зависал», не подавая никаких признаков жизни. Я запрограммировал второй контроллер, вставил вместо первого — то же самое, разве что после этого один нормальный старт стал приходиться не на четыре «зависания», а на три. В общем, если микроконтроллер с отлаженной программой то нормально стартует, то ведет себя кое-как, начните с проверки цепи сброса.

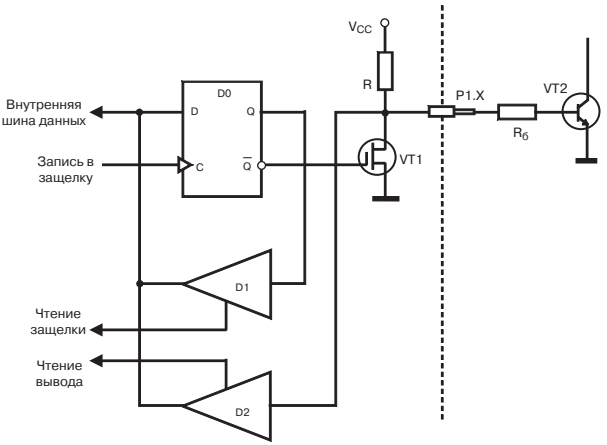
Следующий важный вход — $\overline{EA}$, вывод 31. Если на него подана логическая единица, то микроконтроллер работает с уже упоминавшейся памятью программ, расположенной на кристалле. Нуль на входе заставит микроконтроллер выполнять программу из внешней памяти (такое возможно). О том, как организовывается связь между микроконтроллером 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 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 выводе 30 (ALE) обычно присутствует непрерывная последовательность прямоугольных импульсов с частотой, в 6 раз ниже, чем у кварцевого резонатора, соединенного с выводами 18 и 19. Для 12-мегагерцового кварца она, очевидно, составит 2 МГц. В этой последовательности длительность единицы на выводе ALE примерно вдвое меньше длительности нуля, т. е. скважность составляет 33%. Этот сигнал можно использовать для тактирования микросхем, требующих для работы внешний источник тактового сигнала.

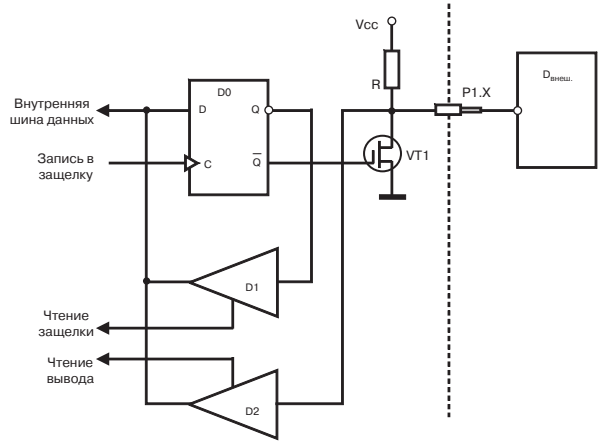
Назначение вывода 29 ($\overline{PSEN}$) будет рассмотрено в разделе, где мы будем говорить о подключении к микроконтроллеру внешней памяти.

Оставшиеся 32 вывода — это линии ввода/вывода информации. Они сгруппированы по 8 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. 1.3** приведена упрощенная схема одной из линий ввода/вывода порта P1.

Как видно из **Рис. 1.3, а**, вывод микросхемы P1.x соединен со стоком выходного полевого транзистора VT1, «подтянутого» к потенциалу питания при помощи внутреннего нагрузочного резистора R. С этим же выво-



а)



б)

Рис. 1.3. Упрощенная схема одной из линий ввода/вывода порта P1

дом микросхемы соединен вход буфера ввода D2. Если мы присоединим к этому выводу 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, используя, таким образом, выбранную линию в качестве линии вывода информации.

Использование линии в качестве линии ввода информации иллюстрируется **Рис. 1.3, б**. Вывод микроконтроллера, а, следовательно, и вход буфера D2, соединены с выходом микросхемы $D_{внеш}$, состояние которого мы хотим проанализировать (иными словами, «ввести» его в микроконтроллер, 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 выводе 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. 1.3**, 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.

1.3. Двоичные и шестнадцатеричные числа

Приведенной выше информации по цоколевке и назначению выводов вполне достаточно для рассмотрения первых примеров. Однако прежде, чем перейти к ним, нам необходимо познакомиться с некоторыми особенностями записи программ и чисел.

При написании программ нам с вами придется пользоваться так называемыми двоичным и шестнадцатеричным представлением чисел. Поначалу привыкнуть к ним довольно трудно. Но после привыкания числа в двоичном и шестнадцатеричном представлении дают вам при анализе и написании программ гораздо больше информации, чем привычные десятичные числа.

В двоичном представлении числа записываются при помощи всего двух цифр — 0 и 1. ***Числам 0 и 1*** в двоичном представлении соответствуют, как и обычно, ***цифры 0 и 1***. А вот с двойкой уже иначе — ***число 2*** в двоичной системе записывается как ***10b*** (буква b на конце служит признаком того, что число записано в двоичном представлении; просто 10 без буквы на конце или 10d — это десять, а 10b — это двойка).

Почему двойка в двоичном представлении записывается таким образом? Да вот почему. В обычной арифметике для первых десяти чисел (от нуля до девяти) есть десять цифр — 0, 1, 2, ..., 9. Для следующего числа, десятки, самостоятельной цифры уже нет. Поэтому для нее мы снова используем самую младшую цифру 0, но ставим слева перед ней цифру 1 — число становится двузначным. Так и с двойкой в двоичном представлении — на нуль и единицу есть свои цифры, а на двойку уже нет (напомню, есть только цифры 0 и1). Поэтому, как и десятка в обычном представлении, двойка в двоичном представлении записывается при помощи младшей из двух возможных цифр— 0, слева перед которым ставим 1.

Тройка в двоичной системе представляется как 11b — это очевидно, 11 — это число, на 1 больше, чем 10. А вот число четыре двумя цифрами в двоичной системе не представить. Действительно, 00b (или 0b) соответствует нулю, 01b (или 1b) соответствует единице, 10b соответствует двойке, 11b — тройке, а пятого двузначного числа, используя только цифры 0 и 1, записать нельзя. Как быть? Элементарно. Коль скоро все двузначные числа кончились, четверка будет трехзначной, причем правые две цифры должны быть нулями, а крайняя слева — 1. Т. е. четыре в двоичном представлении — это 100b, пять — соответственно 101b, шесть — 110b, семь — 111b. Для восьмерки и трех цифр уже не хватает— значит, она должна быть четырехзначной, три правых цифры — нули, крайняя слева какая? Правильно, 1. Итого, восемь — это 1000b.

Надеюсь, принцип представления чисел в двоичной системе я вам объяснил. Если нет — вам придется обратиться к примерам, приведенным в приложении 1 (см. том 2). Ну, а чтобы легко переводить числа из двоичного представления в десятичное и наоборот, проще всего использовать стандартную программу Калькулятор из Windows (кнопка «Пуск» — меню «Программы» — меню «Стандартные» — «Калькулятор»). Запустив «Калькулятор», щелкните мышью заголовок меню «Вид», выберите «Инженерный». Слева над цифрами вы увидите форму представления числа — Hex (шестнадцатеричное, это у нас впереди), Dec (обычное десятичное, оно всегда выбрано при первом запуске программы Калькулятор), Oct (восьмеричное, мы им не будем пользоваться) и Bin — двоичное или бинарное (отсюда и буква b на конце обозначения двоичных цифр). Выбрав Dec — десятичное и набрав 8, кликните мышкой на Bin и вы увидите на экране 1000 (не забудьте, что это — 1000 в системе Bin, т.е. 1000b). Не выходя из

системы Bin, наберите 11000011 и перейдите в Dec — вы увидите на экране 195, т. е. числу 11000011 в двоичной системе соответствует 195 в привычной нам десятичной. И так далее...

В шестнадцатеричном представлении числа записываются при помощи 16 цифр. Десять из них вам хорошо знакомы — 0, 1, 2, ..., 9. В качестве остальных шести цифр используют (не удивляйтесь!) буквы A, B, C, D, E, F. То есть ***для числа десять*** есть своя ***цифра (A)***, для числа ***одиннадцать — цифра B***, для числа ***двенадцать — цифра C***, для числа ***тринадцать — цифра D***. Числам ***четырнадцать*** и ***пятнадцать*** соответствуют, как вы уже догадались, ***цифры E*** и ***F***. А вот для числа шестнадцать цифры уже нет, поэтому его, как и двойку в двоичном представлении, запишем в виде младшей цифры — 0, перед которой поставим цифру 1: таким образом, число ***шестнадцать*** — это ***10h***. Буква h в конце обозначает, что число записано в шестнадцатеричной системе счисления (Hex). Переводить числа из нее в десятичную систему и обратно можно с помощью все того же Windows-Калькулятора. Например, 112Bh — это 4395 десятичное, а 8190 десятичное — это 1FFEh (не поленитесь, проверьте!).

Зачем все это нужно? Так, память программ микроконтроллера AT89C1051 содержит 1024 ячейки, AT89C2051 содержит 2048 ячеек, AT89C51 — 4096 ячеек, а AT89C52 — 8192 ячейки. В шестнадцатеричном представлении это будет соответственно 400h, 800h, 1000h, 2000h. Для процессора 8086, на котором собирались первые IBM-PC, допустимо использование памяти объемом 1048576 ячеек или 100000h. Согласитесь, что в шестнадцатеричном представлении приведенные цифры легче запоминаемы.

А теперь вернемся чуть-чуть назад, к фрагменту памяти программ, который мы анализировали в первом разделе. Обычно адреса ячеек и коды команд представляют именно в шестнадцатеричной, а не в десятичной системе счисления. В Hex-представлении этот фрагмент будет выглядеть так, как показано на **Рис. 1.4**.

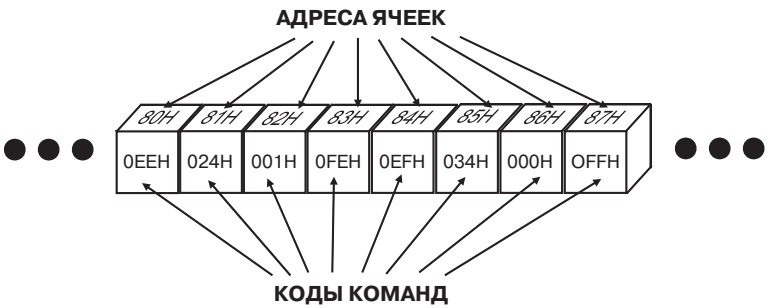


Рис. 1.4. Пример содержимого последовательно расположенных восьми ячеек памяти программ микроконтроллера