УДК 004.431.4
ББК 32.973.26-018.1
Р32

Ревич Ю. В.

Р32 Программирование микроконтроллеров AVR: от Arduino к ассемблеру. — СПб.: БХВ-Петербург, 2020. — 448 с.: ил. — (Электроника)

ISBN 978-5-9775-4076-6

Рассмотрено практическое программирование микроконтроллеров AVR, в том числе популярной платформы Arduino. Рассказано, как выйти за рамки ограничений Arduino, когда следует применять прямое программирование на ассемблере, а когда использовать языки высокого уровня.

Изложены общие принципы устройства микроконтроллеров AVR и их программирования, система команд, программирование таймеров, арифметические операции, память, интерфейсы, режимы энергосбережения и сторожевой таймер, программы реального времени, обмен данными с персональным компьютером. Особое внимание уделено переносу типичных Arduino-проектов на ассемблер.

Даны готовые рецепты для программирования большинства основных функций современной микроэлектронной аппаратуры.

Для учащихся, инженерно-технических работников и радиолюбителей

УДК 004.431.4
ББК 32.973.26-018.1

Группа подготовки издания:

Руководитель проекта	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Сависте*
Компьютерная верстка	*Ольги Сергиенко*
Дизайн серии	*Марины Дамбиевой*
Оформление обложки	*Карины Соловьевой*

Подписано в печать 07.04.20.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 36,12.
Тираж 1000 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Отпечатано в ОАО "Можайский полиграфический комбинат",
143200, г. Можайск, ул. Мира, д. 93

ISBN 978-5-9775-4076-6

Оглавление

ВВЕДЕНИЕ

Почему ассемблер?

По статистике на 2018 год 8-разрядные чипы, в том числе семейство AVR, все еще занимают около 12–13% рынка универсальных контроллеров, что совсем немало, учитывая его миллиардные объемы в количественном исчислении. Дешевые, неприхотливые, несложные в программировании и схемотехнике, 8-разрядные контроллеры с большим количеством задач справляются собственными силами, без дорогих дополнительных компонентов. Вопрос в том, как проще всего приспособить эти контроллеры под свои все расширяющиеся нужды?

Вы уже познакомились с Arduino, имеете представление о возможностях AVR-контроллеров. Изначально рассчитанная на людей, не имеющих инженерного образования, но склонных мастерить нужные вещи своими руками, платформа Arduino в несколько лет завоевала заслуженную популярность любителей по всему миру, породив целую отрасль индустрии. Платы и среда программирования, распространяющиеся под свободной лицензией, породили огромное количество подражаний и ответвлений, развивающих платформу в ту или иную сторону. Одновременно неисчислимое количество фирм по всему миру наладили выпуск самой разнообразной периферии: датчиков, исполнительных устройств, устройств сопряжения с коммуникационными сетями. Наверное, не осталось такой любительской задачи в области конструирования электронных устройств, которую нельзя было бы решить в рамках Arduino, причем чаще всего несколькими способами.

Вероятно, важнейшая особенность экосистемы Arduino, которая и позволила завоевать ей такую популярность — то, что порог вхождения (т. е. сумма необходимых априорных знаний и умений для начала работы) здесь сведен к достижимому минимуму. Можно не разбираться ни в науке об электричестве, ни в программировании — и тем не менее создавать своими руками вполне работоспособные схемы.

Но даже непрофессионалу ясно, что создавая столь удобный инструмент, пришлось чем-то серьезно пожертвовать. Ограничения Arduino начинают чувствоваться сразу, как только вы выходите за рамки макета и пытаетесь создать удобный, экономичный и эстетично выглядящий прибор. Закрадывается мысль — а зачем мне в нем столько всего лишнего, которое потребляет ток, занимает кучу места и после закачки отлаженной программы в контроллер уже никак не используется?

Эта книга для тех, кому надоело раз за разом повторять готовые рецепты из Интернета, закачивая в громоздкую «этажерку» плат готовые решения. Конечно, вам тут же предлагают перейти на нормальный С и работать напрямую с AVR Studio и другими подобными монстрами. И не подумайте, что я вас от этого пути буду отговаривать — смотря, какие задачи вы перед собой ставите. Конечно, на «чистом» С программировать контроллеры удобнее, никто не спорит, и стать профессионалом можно только на этом пути. И переносится такой код при смене модели контроллера даже в рамках одного только семейства AVR лучше, и значительную часть операций проводить проще, текст программы получается компактнее.

> ***ЗАМЕТКИ НА ПОЛЯХ***
>
> Многие любители соблазняются средами программирования microPascal для AVR и BASCom, в которых они могут перейти на знакомые еще по школьным урокам информатики языки Pascal или Basic. Вот этого действительно делать не следует — и не потому, что код получается гораздо хуже, чем в случае С, а потому что так вы еще дальше уходите от сути происходящего внутри контроллера, ничего не приобретая взамен, кроме сиюминутного удобства. Значительная часть преимуществ языка С кроется в доступности готовых библиотек на все случаи жизни, а в случае BASCom и тем более microPascal подобной развитой экосистемы даже близко не сложилось. Из этих сред вам почти некуда развиваться, а в случае С вы, как минимум, приобретаете базу для дальнейшего роста.

Но даже профессионалы вряд ли будут спорить с тем, что имея за плечами только Arduino, с полпинка вы на С ничего хорошего не напишете. Чтобы писать эффективные программы в условиях тотального дефицита ресурсов, надо знать устрашающее количество различных тонкостей. Знания того, что означают, например, термины *static* и *volatile*, недостаточно — надо отчетливо понимать детали механизма действия этих и других спецификаторов в различных случаях, и еще много чего сверх того. А вы твердо уверены, что именно этого хотели — углубиться в изучение тонкостей программирования?

Все споры о преимуществах того или иного подхода обычно ведутся во вполне виртуальном пространстве абстракций, игнорирующих собственно поставленную задачу. Как выразился один знаток программирования, «*одна из сторон доказывает, что дерево плавает, а другая, что кирпич тонет. Естественно, что при такой постановке каждая из сторон уверена в своей правоте и будет вечно ее отстаивать*». Мы постарается не ввязываться в беспочвенные холивары, а сразу сформулируем задачу: мы собираемся создать экономичный, эстетично выглядящий и удобный в применении прибор. Надо ли для этого преодолевать многочисленные нюансы реализации языка С для 8-разрядных контроллеров?

Да, если вы хотите посвятить этому всю жизнь. Но и в этом случае профессионалы (см., например, [2]) рекомендуют начинать с ассемблера, потому что только так можно полностью освоить возможности контроллера. Тем, кто готовится программировать контроллеры профессионально, тоже будет небезынтересно узнать, как в 150 байт кода вмещается программка генерации вполне качественного синуса частотой 50 Гц, способная управлять инвертором напряжения киловаттной мощности (см. *главу 9*).

Заметки на полях

Тут следует еще упомянуть исторический факт, что вообще-то языки высокого уровня, включая C, создавались не для контроллеров гарвардской архитектуры, к которым относятся в том числе и AVR. Для архитектуры фон Неймана, на которую они рассчитаны, принципы составления программ существенно другие (иная парадигма, как говорят программисты). Да, с помощью C можно получить код, почти столь же эффективный, как ассемблерный. Но даже в самом идеальном случае эффективность кода на C будет все-таки только «почти» такая же: живой пример этого — практически все программы этой книги, которые при переводе на C дадут код, заведомо медленнее выполняющийся и большего объема. Специальный пример сравнения программ на Arduino, «чистом» C и ассемблере мы продемонстрируем в *главе 6*.

Изучение собственно AVR-ассемблера, в отличие от C, занимает минимум времени. Краткое описание от Atmel когда-то занимало ровно 4 странички (сейчас оно существенно раздулось [4], но в основном из-за пояснений в виде примеров кода). Его вариант, переведенный на русский, также дополненный примерами и таблицей основных команд, укладывается в 17 страниц [3]. Не сравнить с многостраничными фолиантами руководств по языку C, правда? А почему так? А потому что в элементарном ассемблере специфических для программирования деталей — раз, два и обчелся. Все остальное — структура контроллера, т. е. чистая электроника. И изучать возможности контроллера, когда все время идет речь об установках тех или иных битов в различных регистрах, в терминах команд ассемблера оказывается намного проще и нагляднее.

Мы в этой книге обсудим многие существенные моменты, которые обычно обходятся не только в отношении любительского Arduino, но и часто упускаются из виду профессионалами: как сделать так, чтобы контроллер тратил на операцию минимальное время? Как по максимуму использовать все, что может дать 10-разрядный АЦП? Как снизить потребление схемы до оптимального уровня и правильно настроить режим *sleep*? И постараемся не забывать, что программирование — это всего лишь такой универсальный способ заставить работать схему по заданному алгоритму, а хорошо работающий прибор — это в первую очередь его схемотехника.

Подробности

При изучении материала этой книги читатель, оценивший компактность и предсказуемость ассемблерных программ, но не желающий отказываться от удобств языка C, рано или поздно задаст вопрос: а нельзя ли объединить то и другое в одном проекте? Компилятор AVR-GCC это позволяет делать, как минимум, со стороны проекта на C — вставлять в него ассемблерные фрагменты. Причем вызывать их можно даже двумя способами — см. [9], где описан один из них (а внутри статьи есть ссылка на второй, более распространенный способ). Способы применимы в любой среде программирования, основанной на AVR-GCC, в том числе и в Arduino (правда, лаконичность чисто ассемблерных программ и возможность их составления в любом текстовом редакторе вы при этом теряете). Обратной возможности — использования C-библиотек в ассемблерном тексте, как это позволяют «большие» ассемблеры для ПК (см. [13]), AVR-ассемблер, к сожалению, не предоставляет.

За базовый контроллер мы возьмем традиционный ATmega8 — упрощенную версию ардуиновского ATmega328, которая, однако, умеет почти все то же самое. Две другие базовые модели, программы для которых приведены в этой книге:

ATmega8535 (с бóльшим числом выводов) и ATtiny2313 (с меньшим). Интересно, что ATtiny2313, по идее относящийся к младшему семейству, обладает некоторыми функциями более современных моделей. Эти три модели мы будем считать «нашими». Почти без изменений программы, которые вы здесь встретите, пригодны и для ATmega16, который, если не считать большего объема памяти (совершенно для нас не существенного), в общем, отличается от ATmega8 только увеличенным числом выводов. Особенности переноса программ и на более «продвинутые», и на более простые модели AVR мы уточним по ходу дела.

В первую очередь мы узнаем, что гонять контроллер на частоте 16 МГц совершенно необязательно — для огромного большинства задач достаточно куда более скромной частоты с одновременным снижением потребления в разы. Научимся устанавливать разные режимы контроллера, в том числе от встроенного тактового генератора, когда ему вовсе не нужны никакие дополнительные компоненты. Не обойдем мы и ограничения ассемблера — вы сами сможете нащупать порог, когда более высокое качество ассемблерных программ уже перестает окупаться затратами времени на их создание.

Схемы у нас будут только принципиальные. Картинки монтажной платы с расположением деталей, которые можно так красиво делать в пакете Frizing, только запутывают, потому что самое главное на них отсутствует, и никакой реально необходимой информации о схеме эти картинки не дают. Только принципиальная схема с указанием названий, номиналов, типов и полярностей всех компонентов может дать исчерпывающее представление о том, что вы делаете. Но ничто не мешает сделать ее более наглядной — даже ГОСТ не запрещает располагать компоненты в соответствии с реальной разводкой выводов, делая схему как бы частично монтажной. Именно в таком стиле исполнены все схемы в этой книге. Некоторые элементарные правила, касающиеся обозначений на принципиальных схемах, приведены в *приложении 1*, а интересующихся подробностями отсылаем к моей книге [10].

Автор предполагает, что читатель худо-бедно знаком с двоичными и шестнадцатеричными цифрами и основными логическими операциями, умеет читать принципиальные схемы и знает, как связаны напряжение и ток, и зачем светодиоду нужен резистор. Тем, кто в таких вопросах плавает, адресован краткий «Ликбез» (см. *приложение 1*), а также совет сначала ознакомиться с книжкой [10], где все эти вопросы разъясняются детально. Таблицы с основными характеристиками некоторых моделей микроконтроллеров Atmel AVR из числа самых ходовых приведены в *приложении 2*.

Обсуждений радиолюбительских технологий в этой книге вы также почти не встретите — предполагается, что все примеры и макеты конкретных конструкций выполняются на беспаечной макетной плате, а особенности переноса их в законченные устройства и нюансы подбора компонентов обговариваются только в случаях, когда это критично с точки зрения функциональности.

Книга построена по традиционному принципу «от общего к частному»: в *части 1* рассматриваются общие вопросы устройства контроллеров, определяются необхо-

димые приспособления и инструменты, описаны программные средства и их настройка. К практическому программированию мы переходим в *части II*, которая начинается с простейших примеров ассемблерных программ и краткого обзора системы команд. В *части III* изучение продолжается примерами использования конкретных узлов контроллеров.

В архиве, который вы можете скачать по адресу **http://revich.lib.ru/AVR/AVR-asm.zip**, вы найдете тексты и примеры большинства программ, размещенных в этой книге, распределенные по папкам с названием главы. В тексте глав указываются имена файлов, соответствующих этим программам. Все включенные в книгу схемы, программы и отдельные алгоритмы проверены на макетах, и их работоспособность гарантируется при соблюдении указанных для них условий.

Связаться с автором можно по электронной почте **revich@lib.ru**, можно также оставить личное сообщение на сайте **habr.com** пользователю **YRevich**.

ЧАСТЬ I

Общие принципы устройства и функционирования Atmel AVR

ГЛАВА 1

Обзор микроконтроллеров AVR

Говорят, что в 1960-е годы, наблюдая за участниками студенческих демонстраций протеста, Гордон Мур заметил: «Истинные революционеры — это мы». Ученик и сотрудник одного из изобретателей транзистора У. Шокли, в числе прочего считающегося основателем знаменитой Кремниевой долины, в свою очередь основатель и лидер компаний, которым суждено было сыграть ведущую роль в развитии микроэлектроники, Мур знал, что говорил. Парадоксальным образом именно изобретениям Мура и его сотрудников было суждено стать основой того мира, в котором впоследствии сконцентрировалась деятельность «бунтующей молодежи» 1960-х. Современные хакеры (не компьютерные хулиганы из новостей, а настоящие увлеченные своим делом компьютерщики) — прямые идеологические наследники сорбоннских студентов и американских демонстрантов, сменившие девиз «Make love not war»[1] на «Не пишите лозунги — пишите код». Неслучайно многие известные деятели электронно-компьютерной индустрии, авторы изобретений, сформировавших лицо современного мира, — выходцы из среды, близкой той самой «бунтующей молодежи».

Наша история о микроконтроллерах началась с того, что в 1957 году Гордон Мур совместно с Робертом Нойсом и еще шестью сотрудниками Shockley Semiconductor Labs (Шокли назвал их «предательской восьмеркой») основали компанию Fairchild Semiconductor. Ей мы обязаны не только развитием полупроводникового рынка и внедрением микросхем в инженерную практику, но и тем, что она стала своеобразной кузницей кадров и генератором идей для молодой отрасли. Из изобретений сотрудников Fairchild берет свои истоки большая часть ключевых инноваций, сформировавших лицо современного мира, начиная прямо с самой основы — изобретения микросхемы Робертом Нойсом.

Из всего урагана событий, источником которых стала Fairchild, для нашего повествования важно то, что в числе прочих инноваций сотрудники Fairchild первыми стали продвигать полупроводниковую память. Сейчас, в век CD и DVD, твердотельных жестких дисков и flash-карточек, нам трудно представить себе, что в нача-

---

[1] «Занимайтесь любовью, а не войной» — лозунг хиппи 1960-х, протестующих против войны во Вьетнаме.

ле 1960-х годов программы для компьютеров хранились в основном на картонных листочках (перфокартах), конструкторы ломали голову над дорогущими модулями ОЗУ на ртутных линиях задержки, осциллографических трубках и ферритовых колечках в полмиллиметра диаметром, где каждый бит «прошивался» вручную. Самое компактное в те годы электронное устройство для хранения данных на магнитных дисках под названием RAMAC 305 емкостью 5 Мбайт было размером с промышленный холодильник и сдавалось в аренду за 5 тыс. долларов в месяц.

Компактная полупроводниковая память была нужна абсолютно всем — от военных и NASA до изготовителей бытовых приборов. Почуяв, откуда дует ветер, в 1968 году Мур с Нойсом оставили Fairchild и основали Intel, как специализированную компанию по разработке и производству памяти. Они еще не ведали, что самым популярным детищем Intel станет вовсе не память, а микропроцессор, разработка которого первоначально затевалась, как вспомогательный этап в проектировании обычного калькулятора.

С набора из четырех небольших микросхем Intel под названием «семейство 4000» началось победное шествие микропроцессоров по всему миру. Они весьма быстро разделились на несколько разновидностей, в основном относящихся к двум главным группам: собственно микропроцессорам (МП) и микроконтроллерам (МК). Первые предназначены для использования в составе вычислительных систем, самые распространенные из которых — персональные компьютеры (ПК), поэтому их еще часто называют «процессорами для ПК» (к этой же группе обычно относят также и производительные МП для серверов и некоторые другие). МК отличаются от МП тем, что они в первую очередь предназначены для управления различными системами, поэтому при относительно более слабом вычислительном ядре они включают в себя много дополнительных узлов. То, что для обычного МП предполагается размещать во внешних «чипсетах» или дополнительных модулях (память, порты ввода/вывода, таймеры, контроллеры прерываний, узлы для обработки аналоговых сигналов и пр.), в МК располагается прямо на кристалле, отчего их часто называют «computer-on-chip» («однокристальный компьютер»).

И действительно, в простейшем случае для построения полностью функционирующего компьютера достаточно единственной микросхемы МК с подсоединенными к ней устройствами ввода/вывода. Современные модели рядовых однокристальных МК превышают вычислительные возможности IBM PC AT на 286-м процессоре образца второй половины 1980-х. Есть быстроразвивающиеся области, где границу между МП и МК провести трудно — таковы, например, процессоры для мобильных устройств, от обычных телефонов до смартфонов и планшетов, в которых процессорный узел должен обладать развитыми вычислительными функциями и управлять многочисленными внешними компонентами.

AVR и другие

Собственно история AVR-контроллеров началась с того, что 1962 году в Калифорнии появилась семья Перлегос, греческих эмигрантов, уроженцев города Триполис. Родители занялись, как и на родине, виноградарством, а сыновья Джордж и Гюст

Перлегос выбрали модную специальность инженера-электронщика — оба окончили вначале университет Сан-Хозе, а затем Стэнфордский университет. В 1974 году в возрасте 24 лет младший из братьев Джордж Перлегос начал работать в компании Intel, где попал на одно из самых передовых направлений — разработку электрически стираемой памяти для замены «прожигаемой» OTP ROM. При его участии, а затем и под его непосредственным руководством были созданы две технологии, ставшие точкой роста для всей отрасли по производству flash-памяти — одного из главных столпов современной «цифровой революции».

В начале 1980-х Джордж Перлегос увольняется из Intel. С братом Гюстом и еще несколькими сотрудниками он в 1984 году создает вскладчину на личные средства компанию, полное название которой звучит как Advanced Technology MEmory and Logic или сокращенно — Atmel.

Сначала продукцией Atmel были микросхемы энергонезависимой памяти всех разновидностей: как «однократно программируемых» OTP EPROM и «перезаписываемых» EEPROM с последовательным и параллельным доступом, так и Flash. В 1985 году Atmel выпустила первую в мире EEPROM по доминирующей ныне КМОП-технологии, а в 1989 году — первую flash-память с питанием от одного напряжения +5 В. В конце 1980-х Intel вознамерилась наказать ряд компаний-производителей EPROM, в том числе и Atmel, якобы за нарушение патентов, но в конце концов им удалось договориться об обмене лицензиями. Причем в конечном итоге Atmel перепала лицензия на производство классического микроконтроллера 8051, от поддержки которого Intel уже в то время постепенно отходила, сосредоточившись на процессорах для ПК.

Подробности

Напомним, что EEPROM отличается от flash-памяти тем, что первая допускает раздельный доступ к любой произвольной ячейке, а вторая — лишь к целым блокам. Поэтому EEPROM меньше по объему (характерный объем специализированных микросхем EEPROM — от единиц килобит до единиц мегабит) и дороже, в настоящее время ее используют в основном для хранения данных, в том числе в составе микроконтроллеров. Flash-память проще и дешевле, и к тому же дает значительный выигрыш в скорости при больших объемах информации, особенно при потоковом чтении/записи, характерном для медиаустройств (вроде цифровых камер или MP3-плееров). В составе микроконтроллеров flash-память служит для хранения программ. Некоторые подробности о различных типах памяти и их функционировании приведены также в *главе 10*.

Так Atmel оказалась «втянута» в число производителей микроконтроллеров, в котором очень быстро оказалась на первых позициях, — в 1993 году началось производство первых в отрасли МК AT89C51 со встроенной flash-памятью программ. Это означало начало переворота во всей инженерной практике, потому что существовавшие ранее МК обладали либо однократно программируемой OTP-памятью, либо УФ-стираемой, которая значительно дороже в производстве, и работа с ней приводит к большим потерям времени разработчиков. Число циклов перезаписи для УФ ППЗУ не превышает нескольких десятков, а прямой дневной свет, попавший на такой кристалл, может привести к стиранию информации. Поэтому даже мелкосерийные устройства приходилось изготавливать преимущественно с исполь-

зованием OTP ROM, что значительно рискованнее, — изменить в случае даже малейшей ошибки записанную программу уже было невозможно. Появление flash-памяти изменило весь «ландшафт» в этой области — именно в результате ее внедрения стали возможными такие операции, как программное обновление BIOS компьютера или «перепрошивка» управляющих программ для бытовых электронных устройств.

В 1995 году два студента Норвежского университета естественных наук и технологии из города Тронхейма, Альф Боген и Вегард Воллен, выдвинули идею 8-разрядного RISC-ядра, которую предложили руководству Atmel. Имена разработчиков вошли в название архитектуры AVR: Alf + Vergard + RISC. Идея настолько понравилась, что в 1996 году в Тронхейме был основан исследовательский центр Atmel, и уже в конце того же года выпущен первый опытный микроконтроллер новой серии AVR под названием AT90S1200. Во второй половине 1997 года корпорация Atmel приступила к серийному производству семейства AVR.

Почему AVR?

У AVR-контроллеров «с рождения» есть две особенности, которые отличают это семейство от остальных 8-разрядных МК. Во-первых, это наличие конвейера, благодаря чему для AVR не существует понятия машинного цикла: большинство команд выполняются за один такт. Для сравнения отметим, что пользующиеся большой популярностью МК семейства PIC выполняют команду за 4 такта, а классические 8051 — вообще за 12 или даже 24 такта.

Вторая особенность — наличие 32 оперативных регистров, не совсем равноправных, но позволяющих в ряде случаев вообще не обращаться к оперативной памяти и не использовать в явном виде стек (что в принципе невозможно в том же семействе *x*51). Более того, в младших моделях Tiny AVR стек вообще недоступен для программиста. Потому структура ассемблерных программ для AVR стала подозрительно напоминать программы на языке высокого уровня, где операторы взаимодействуют не с ячейками памяти и регистрами, а с абстрактными переменными и константами. Программы при этом работают много быстрее и с точно предсказуемым временем выполнения отдельных операций.

Суммировав мнения из различных источников и опираясь на собственный опыт, автор пришел примерно к такому подразделению областей применения самых распространенных семейств 8-разрядных контроллеров:

- контроллеры классической архитектуры *x*51 (первые микросхемы семейства 8051 были выпущены еще в начале 1980-х) уже давно не применяются на практике и лучше всего подходят для общего изучения предмета — по аналогии с языками Pascal или Basic при обучении программированию. Intel-ассемблер замечательно продуман, программы на нем хорошо читаются, число различных команд для *x*51 весьма велико (*x*51, в отличие от многих других, имеет не RISC-, а CISC-архитектуру, для которой характерно наличие большого числа весьма функциональных команд). Авторитетный Di Halt [2], отмечал: «*Что касается лично меня, то я обожаю архитектуру 51 за ее чертовски приятный ассемблер,*

на котором просто кайфово писать. Под эту архитектуру уже написаны гигабайты кода, созданы все мыслимые и немыслимые алгоритмы»;

- семейство AVR рекомендуется для начинающих электронщиков-практиков — в силу универсальности устройства, легкости загрузки программ, преемственности структуры для различных типов контроллеров, разнообразия типов корпусов, простоты схемотехники, практически лишенной каких-либо специфических особенностей, затрудняющих освоение новичками. Отдельно следует отметить наличие отличной базы для начала работы с этими МК в виде платформы Arduino и всего с ней связанного (подробнее о платформе Arduino рассказано чуть далее);
- контроллеры PIC фирмы Microchip не имеют столь удобной системы команд (которых всего около трех десятков), отдельные представители семейства не универсальны, и требуют тщательного выбора «под задачу». Зато у них низкое энергопотребление и быстрый старт. PIC идеально подходят для проектирования несложных устройств, особенно предназначенных для тиражирования. Традиционно они используются в «умных» узлах автомобилей, а также в устройствах бытовой сигнализации;
- последние годы набирает обороты семейство STM8, впервые выпущенное фирмой STMicroelectronics в 2008 году. STM8 в целом похожи на AVR и PIC, но отличаются как от них, так и от своих старших собратьев — STM32, в сторону некоторых удобств: большего диапазона питания, большей скорости выполнения команд за счет большей глубины конвейера, большей тактовой частоты при меньшем потреблении, мелкими усовершенствованиями периферии ядра и т. п. Но главное то, что наученные горьким опытом своих коллег из других фирм разработчики архитектуры STM8 сосредоточились на максимальной совместимости всех контроллеров семейства между собой. AVR в этом смысле далеко не идеальны (хотя, заметим, все-таки получше других семейств), но STM8 превзошли всех, заявив совместимость по выводам — одни и те же узлы в разных контроллерах выводятся на одни и те же выводы корпуса, что позволяет менять один контроллер на другой без переделки кода и даже без переделки платы.

  Свихнувшимся на простейших инструментах адресована ST Visual Develop — фирменная среда разработки, позволяющая в том числе программировать на ассемблере STM8, куда более простая и в установке, и в освоении, чем монструозная Atmel Studio. Правда, архитектура STM8 намного лучше AVR приспособлена к языку C, и там использование ассемблера почти теряет смысл. Возможно, единственное препятствие для распространения STM8 в широких кругах любителей — то, что STM8 выпускаются исключительно в совместимых друг с другом планарных корпусах с мельчайшим шагом 0,5–0,6 мм и потому трудно поддаются макетированию и пайке «на коленке».

Как ни странно, но 8-разрядные контроллеры вопреки предсказаниям десятилетней давности не стали исчезать под натиском 32-разрядных чипов: в конце предыдущего десятилетия они составляли более 50% всех продаваемых микропроцессорных изделий, а сейчас их доля сократилась до 12–13%, но вот уже несколько лет остает-

ся постоянной. Это понятно — зачем нужен высокопроизводительный 32-разрядный процессор для управления стиральной машиной? Несмотря на сопоставимую цену 32-разрядных моделей, вследствие более простой схемотехники, а также экономии времени на программировании и отладке, популярность 8-разрядных контроллеров не падает.

Об Arduino

Популярность как любительского конструирования электронных приборов вообще, так и конкретно AVR-микроконтроллеров в среде любителей существенно возросла с появлением платформы Arduino. Платформа возникла в среде сотрудников Interaction Design Institute (что можно перевести как «Институт конструирования взаимодействий») из итальянского городка Ивреа и получила свое почти толкиеновское название по имени реально существовавшего короля Ардуина, правившего этой местностью в начале прошлого тысячелетия. Arduino выросла из задачи научить студентов непрофильных специальностей создавать электронные устройства, причем быстро и, желательно, без опоры на углубленное изучение электроники, электротехники и программирования.

В конце концов группа, руководимая программистом Массимо Банци, создала универсальную аппаратную платформу на основе дешевых, удобных и доступных микроконтроллеров Atmel AVR и решила ее распространять на принципах open source. Такие свободные лицензии, как знаменитая GPL, разработанная применительно к софту, для «железа» напрямую не годится, потому создатели взяли за основу пакет лицензий Creative Commons для творческих продуктов. Лицензия Arduino запрещает использование этой торговой марки для каких бы то ни было сторонних продуктов, кроме расширений основного проекта. Это привело к тому, что от Arduino стали отпочковываться аналогичные проекты, совместимые с ним, но желающие иметь иные названия — например, такие, как Freeduino, Craftduino, Carduino и многие другие.

Пик роста возможностей Arduino приходится на конец нулевых — начало десятых годов XXI века, потом темпы развития платформы снизились. Разработчики поневоле загнали себя в тупик — ограничив выбор контроллеров двумя-тремя моделями и стандартизировав дизайн плат, они сильно облегчили порог вхождения пользователям-непрофессионалам и производителям совместимого оборудования, но своими руками закрыли себе пути совершенствования. Отсюда и интерес к расширенному использованию контроллеров AVR, которое в том числе призвана утолить эта книга. Но скорее всего Arduino еще долго будет доминировать в любительском секторе — пока кто-нибудь не придумает что-нибудь столь же простое и удобное, только уже, наверное, на 32-разрядной платформе.

Примерно то же самое, что и с Arduino, происходит с самими AVR-контроллерами. В руководстве Atmel «отец flash-памяти» и лауреат многих отраслевых наград Джордж Перлегос пребывал до 2006 года, когда оставил свой пост. Через десять лет, в 2016-м, компания была поглощена фирмой Mirochip. За это десятилетие в руководстве не нашлось инициативного человека, способного предложить идею кардинального обновления модельного ряда так, чтобы и преемственность сохранить, и обеспечить рост на долгий период (подобно тому, как это сделали в STMicroelectronics, предложив STM8 вместо мало кому интересных STM7). Но это, конечно, не значит, что AVR вымирают — достаточно посмотреть на обширный список моделей на сайте Microchip, напротив которых стоит «In Production», причем не такая уж маленькая часть этого списка входит в категорию «New product». Для AVR еще на долгие годы обеспечена своя ниша, и уж в любительском секторе даже признаков увядания семейства пока не наблюдается.

Краткий обзор возможностей AVR

Atmel AVR представляет собой семейство универсальных 8-разрядных микроконтроллеров на основе общего ядра с различными встроенными периферийными устройствами. Возможности МК AVR позволяют решить множество типовых задач, возникающих перед разработчиками радиоэлектронной аппаратуры.

Особенности микроконтроллеров Atmel AVR:

- **Производительность порядка 1 MIPS/МГц**, где MIPS (Millions of Instructions Per Second, миллион команд в секунду) — одна из самых старых и во многом формальная характеристика производительности процессоров, т. к. наборы команд для различных процессоров различаются, и, соответственно, одно и то же число инструкций на различных системах даст разную полезную работу. Тем не менее для простых 8-разрядных вычислительных систем, не содержащих команд, оперирующих с большими числами, числами с плавающей точкой и массивами данных, это неплохой показатель для сравнения их производительности. Вычислительное ядро AVR на ряде задач по производительности превосходит 16-разрядный процессор 80286.
- **Усовершенствованная RISC-архитектура** — концепция RISC (Reduced Instruction Set Computing, вычисления с сокращенным набором команд) предполагает наличие набора команд, состоящего из минимума компактных и быстро выполняющихся инструкций. При этом такие более громоздкие операции, как вычисления с плавающей точкой или арифметические действия с многоразрядными числами, предполагается реализовать на уровне подпрограмм. Концепция RISC упрощает устройство ядра (в типовом ядре AVR содержится лишь 32 тыс. транзисторов, в отличие от десятков миллионов в процессорах для ПК) и ускоряет его работу — типовая инструкция выполняется за один такт, кроме команд ветвления программы, обращения к памяти и некоторых других, оперирующих с данными большой длины. В AVR имеется простейший двухступенчатый конвейер, когда команда выполняется в одном такте с выборкой следующей. В отличие от Intel-архитектур, в «классическом» AVR не было аппаратного умножения/деления, однако в подсемействе Mega появились операции умножения.
- **Раздельные шины памяти команд и данных** — AVR (как и большинство других микроконтроллеров) имеет т. н. *гарвардскую архитектуру*, где области памяти программ и данных разделены (в отличие от классической архитектуры фон Неймана в обычных компьютерах, где память общая). Раздельные шины для этих областей памяти значительно ускоряют выполнение программы — данные и команды могут выбираться одновременно.
- **32 регистра общего назначения (РОН)** — Atmel была первой компанией, далеко отошедшей от классической модели вычислительного ядра, в которой выполнение команд предусматривает обмен данными между АЛУ и запоминающими ячейками в общей памяти. Введение РОН в таком количестве (напомним, что в архитектуре *x*86 всего четыре таких регистра, а в *x*51 понятие РОН, как таковое, отсутствует) в ряде случаев позволяет вообще отказаться от расположения

глобальных и локальных переменных в ОЗУ и от явного использования стека, операции с которым усложняют и загромождают программу.

Подробности

В ранних архитектурах обращение к памяти занимало много процессорного времени, потому в целях ускорения выполнения арифметических операций их стали производить в специально выделенных регистрах (регистрах общего назначения, РОН), к которым часто добавлялся специальный регистр-аккумулятор. В AVR от единого аккумулятора отказались в пользу большого количества равноправных РОН, что облегчает создание программ непосредственно в инструкциях процессора (на ассемблере), но усложняет и запутывает задачу компилятора с языков высокого уровня. Наконец, в архитектуре STM (и 8-ми и 32-битовых) опять отказались от концепции РОН в пользу единственного аккумулятора — все операции АЛУ производит непосредственно с переменными в оперативной памяти, — но на этот раз они выполняются за один такт. Иными словами, в архитектуре STM вы можете насоздавать сколько угодно аналогов регистров общего назначения — ограничивает только объем памяти. Платой за это в STM8 служит, во-первых, весьма запутанная система распределения памяти, за чем приходится внимательно следить (помните первую компьютерную аксиому: «Закладывая что-то в память компьютера, помните, куда вы это положили»?), во-вторых, некоторая неопределенность с временем выполнения команд перехода из-за наличия трехуровневого конвейера. Зато и по части совместимости с языком С — никаких проблем.

- Большое количество разнообразных **команд** (инструкций), номенклатура которых (примерно от 90 до 130, в зависимости от модели контроллера) для AVR больше, чем в других RISC-семействах. Противоречия с концепцией RISC тут нет, поскольку значительная часть этих инструкций — псевдонимы, и введены они исключительно для удобства программирования. Так что изучать все инструкции сразу не потребуется.
- **Flash-память программ** (10 000 циклов стирание/запись) — с возможностью внутрисистемного перепрограммирования, т. е. загрузки программ прямо в готовой схеме (In System Programming, ISP). ISP не следует путать с программированием через последовательный (Serial) порт с помощью отдельной программы-загрузчика (Bootloader), как это делается в Arduino (подобно об этом рассказано в *главах 5* и *6*).
- **Отдельная область энергонезависимой памяти** (EEPROM, 100 000 циклов стирание/запись) — для хранения данных с возможностью записи программным путем или внешней загрузки через ISP, подобно программам (см. *главу 10*).
- **Встроенные устройства для обработки аналоговых сигналов**: аналоговый компаратор и многоканальный 10-разрядный АЦП (см. *главу 11*).
- **Сторожевой таймер**, позволяющий осуществлять автоматическую перезагрузку контроллера через определенные промежутки времени (например, для выхода из «спящего» режима — см. *главу 14*).
- **Последовательные интерфейсы SPI, TWI (I^2C) и UART (USART)**, позволяющие осуществлять обмен данными с большинством стандартных датчиков и других внешних устройств аппаратными средствами (см. *главы 12*, *13* и *15*).

- **Таймеры-счетчики** с предустановкой и возможностью выбора источника счетных импульсов: как правило, один-два 8-разрядных и как минимум один 16-разрядный, в том числе могущие работать в режиме PWM (ШИМ) — многоканальной 8-, 9-, 10-, 16-битовой широтно-импульсной модуляции (см. *главу 9*).
- **Возможность работы при тактовой частоте** от 0 Гц до 16–20 МГц.
- **Диапазон напряжений питания от 2,7 до 5,5 В** (в некоторых случаях от 1,8 или даже 0,7 В).
- **Многочисленные режимы энергосбережения**, отличающиеся числом узлов, остающихся подключенными. Выход из «спящих» режимов осуществляется по сторожевому таймеру или по внешним прерываниям (см. *главу 14*).
- **Встроенный монитор питания** — детектор падения напряжения (Brown-out Detector).

Здесь упомянуты далеко не все особенности, характерные для различных моделей AVR. С некоторыми другими мы познакомимся в дальнейшем, а также на практике рассмотрим упомянутые подробнее. Но сначала дадим общую характеристику различных семейств AVR с точки зрения их преимущественного назначения.

Семейства и модификации AVR

В 2002 году фирма Atmel начала выпуск новых подсемейств 8-разрядных МК на базе AVR-ядра. С тех пор МК этого семейства стали делиться на три основные группы (подсемейства): Classic, Tiny и Mega. В 2008 году к ним добавилось семейство Xmega, которое популярности не снискало, и потому в этой книге мы его не рассматриваем. МК семейства Classic (AT90S*xxxx*) уже давно не выпускаются — дольше всего в производстве «задержалась» очень удачная (простая, компактная и быстродействующая модель) AT90S2313, но и она была еще в 2005 году заменена на ATtiny2313.

Все «классические» AVR с первыми цифрами 2, 4 и 8 в наименовании модели (что означает количество килобайт памяти программ) первоначально имели полные аналоги в семействах Tiny и Mega, и большая часть таких совместимых моделей выпускается до сих пор. Для Mega-аналогов при программировании возможна установка специального бита совместимости, который позволяет без каких-либо изменений использовать программы, созданные для семейства Classic. Поэтому, если вы найдете на интернет-ресурсах примеры программирования AVR семейства Classic, то их можно будет задействовать в современных моделях без каких-либо доработок. Тем не менее в книге все примеры адаптированы для современных моделей Tiny и Mega.

Микросхемы Tiny в основном имеют Flash-ПЗУ программ объемом 1–8 кбайт (в некоторых современных моделях расширенное до 16 кбайт) и размещаются в корпусах с 6–32 выводами, т. е. они в целом предназначены для более простых и дешевых устройств. Это не значит, что их возможности во всех случаях более ог-

раниченны, чем у семейства Mega. Так, например, ATtiny26 содержит таймер с высокоскоростным ШИМ-режимом, а также 11-канальный АЦП с возможностью работы в дифференциальном режиме с регулируемым входным усилителем и встроенным источником опорного напряжения, что характерно для семейства Mega. Микросхема ATtiny2313, как уже говорилось, представляет собой улучшенную версию одного из наиболее универсальных и удобных «классических» AVR AT90S2313, дополненную возможностями новых семейств. Старшие модели семейства Tiny в настоящий момент по некоторым параметрам даже обходят младшие Mega — например, у ATtiny2313 есть расширенное внешнее прерывание PCINT на почти всех цифровых выводах портов (что дает возможность без излишних сложностей выводить его из режима энергосбережения), а у ATmega8/16 такая возможность отсутствует.

Подсемейство Mega оснащено Flash-ПЗУ программ объемом 4–256 кбайт и имеет корпуса с 28–100 выводами. В целом МК этой группы более «навороченные», чем Tiny, имеют более разветвленную систему встроенных устройств с более развитой функциональностью.

Таблицы с основными характеристиками некоторых моделей Tiny и Mega из числа самых ходовых приведены в *приложении 2*. Там же даны некоторые общие технические характеристики семейства AVR. Таблицы ориентированы на основные нужды радиолюбителей, в основном не выходящие за рамки сведений из последующих глав этой книги, и представляют собой небольшую выборку из фирменной параметрической таблицы 8-разрядных AVR, которую можно найти на сайте **www.microchip.com** (не путать с таблицей Quick Reference Guide, доступной практически с главной страницы сайта). Надо отметить, что пользоваться этой таблицей не слишком удобно — там все модели помещены внавал по непонятному принципу сортировки, а некоторые важные параметры отсутствуют. Поэтому, помимо *приложения 2*, рекомендую еще две более полные (хотя и частично устаревшие) выдержки из фирменной таблицы на русском языке отдельно для Mega и Tiny [5], где ориентироваться гораздо проще.

Более подробные сведения можно почерпнуть из фирменной технической документации, которая доступна на сайте **www.microchip.com** для всех без исключения моделей, включая снятые с производства. Документация размещается в англоязычных файлах в формате PDF, традиционно называемых Datasheets (мы их будем по-свойски именовать «даташитами»), и только там имеется полная официальная информация по каждому контроллеру. Нельзя сказать, что в «даташитах» вовсе не бывает ошибок и недоработок (например, таковые найдены автором в свежей редакции файла по контроллеру ATmega8a, выпущенной уже под лейблом Microchip, а не Atmel), и поэтому в конце каждого такого файла ведется учет «ревизий» предыдущих версий. Но в целом информации из этих документов можно и нужно доверять, и всегда следует сверяться именно с ними в случаях, когда сведения из какого-то постороннего источника вызывают сомнения.

Наиболее полный и скрупулезно выполненный перевод официальных «даташитов» для ряда моделей AVR представляют книги А. В. Евстифеева [6,7]. Книги эти не-

сколько устарели с точки зрения представленного модельного ряда, но все нужные нам модели в них охвачены, и их очень полезно иметь под рукой с другой точки зрения — в этих книгах приведен грамотный и в литературном, и в техническом плане перевод на русский язык не всегда внятно написанной фирменной документации, американский технический язык которой может быть весьма далек от английского литературного, который мы все изучали в школе. Обратите внимание: первые издания этих книг (под маркой издательства «Додэка») решительно отстали от современности по части ассортимента представленных там контроллеров. И если они вдруг у вас имеются, то из них могут пригодиться только переведенные на русский подробные описания команд из официального руководства [8]. Такие же описания команд на русском есть и в более современных изданиях [6,7].

Кроме Datasheets, официальная документация представлена в виде Application Notes, что можно перевести как «замечания к применению». «Аппноты» нумеруются подряд по мере их выхода, без какого-либо отношения к содержанию, и найти в них что-то можно только полнотекстовым поиском. За двадцать с лишним лет существования AVR «аппнот» накопилось на немаленькое собрание сочинений, и значительная часть их содержит описания каких-то не очень интересных частных случаев. Тем не менее в них можно найти и конкретные примеры программирования часто встречающихся задач, и описания построения рекомендуемых алгоритмов, и тонкости различий между родственными моделями контроллеров, и еще много разной полезной информации, так что мы будем периодически на них ссылаться.

Основные принципы маркировки AVR

Все модели каждого семейства могут иметь несколько модификаций. Если не считать некоторых младших моделей с объемом памяти менее 2 кбайт в серии Tiny (Tiny4, 5 и 9), то первые одна или две цифры в наименовании модели сразу после обозначения семейства (ATtiny или ATmega) означают объем flash-памяти программ в килобайтах, так: ATmega8 — это контроллер с памятью программ 8 кбайт, а ATmega16 — контроллер с памятью программ 16 кбайт. Если имеются более современные модификации этих контроллеров, то они обозначаются прибавлением еще одной или нескольких цифр — например, ATmega88 есть расширенная версия ATmega8 (учтите, что это фактически другая ветвь AVR, и без модификации ассемблерные программы, написанные для Mega8, применять к Mega88 не получится). ATmega88, в свою очередь, имеет ряд вариантов с разным объемом памяти программ: ATmega48 (4 кбайта), ATmega168 (16 кбайт), в том числе и наш любимый ардуиновский ATmega328 (32 килобайта).

Буква «L» в обозначении старых моделей говорила о расширенном диапазоне питания 2,7–5,5 В при сниженной допустимой тактовой частоте (8 МГц). Отсутствие такой буквы означало диапазон питания 4,5–5,5 В при повышении допустимой тактовой частоты (16 МГц). В настоящее время появились и другие буквы: например, буква «A» говорит о переходе на новую технологию с уменьшенными технологи-

ческими нормами (60 нм вместо 90), и такой кристалл перекрывает возможности обеих старых версий («L» и не-«L») при еще более уменьшенном потреблении. Буква «V» означает версии контроллеров, работающих при расширенном напряжении питания в пределах 1,8–5,5 вольта, буква «U» — сверхнизкое допустимое питание от 0,7 вольта, буква «P» — понижение потребления в «спящем» режиме (не путать с «P» в обозначении варианта исполнения, о чем см. далее). После этих букв может идти еще одна буква позднейшей модификации — например, недавно вышедшая версия ATmega328PB сильно отличается от оригинальных ATmega328 или ATmega328P.

Поэтому при выборе конкретного типа микросхемы нужно быть внимательным и проверять, что означают те или иные буквы в полном наименовании. Для примера: поскольку L-версии одновременно также и менее быстродействующие, то у большинства из них максимальная тактовая частота ограничена значением 8 МГц, а для «обычных» или A-версий максимальная частота составляет 16 или 20 МГц. Хотя, как правило, при разгоне L-микросхем с напряжением питания 5 В до частот 10–12 МГц неприятностей ожидать не следует (аналогично версии без буквы «L» вполне могут работать при напряжении питания около 3 В, разумеется, не на экстремальных значениях частот), тем не менее при проектировании высоконадежных устройств следует учитывать это требование.

После основного обозначения модели через дефис идет обозначение варианта исполнения. Первые одна или две цифры здесь обозначают максимальную рабочую частоту в мегагерцах, последняя буква — условное обозначение температурного диапазона (чаще всего у нас будет встречаться буквы «U» или «I», означающие индустриальный температурный диапазон от –40°C до +85°C). А вот буква посередине, сразу после частоты, нас будет очень интересовать, потому что она означает тип корпуса (подробнее об этом рассказано в *главе 4*).

Подчеркнем, что с точки зрения внутренней начинки, а следовательно, и программирования, никакой разницы между AVR-контроллерами с разными буквенными индексами нет (упомянутый новый ATmega328PB представляет собой неприятное исключение, так что проверять все-таки надо). Основные различия относятся к потреблению, максимальной рабочей частоте, а также отчасти к схемотехнике — микросхемы в четырехсторонних планарных корпусах TQFP имеют на четыре вывода больше, чем в двухрядных DIP. Внимательное сравнение разводки выводов в разных корпусах, однако, показывает, что никакой особой дополнительной функциональности лишние выводы не несут — дублируются выводы питания Vcc и GND, а также появляется пара дополнительных каналов АЦП. Интересно, что эти дополнительные каналы (ADC6 и ADC7) имеются на некоторых платах Arduino, использующих контроллер в планарном корпусе: например, они выведены на отдельные штырьки на плате Mini, а вот на Arduino Nano отсутствуют — их место занял ISP-разъем.

Кроме этих двух семейств, на базе AVR-ядра выпускаются «навороченные» AVR семейства Xmega, специализированные микросхемы с промышленным интерфей-

сом CAN для управления ЖК-дисплеями, с беспроводным интерфейсом IEEE 802.15.4 (ZigBee) для предприятий торговли, высокотемпературные для использования в автомобильной промышленности (версии Automotive). Из этого обширного перечня нас могут заинтересовать микросхемы с буквой «U» после обозначения типа — это микросхемы со встроенным USB-интерфейсом. Именно на основе такого контроллера (ATmega32U4) спроектирован Arduino Leonardo, отчего он при подключении к компьютеру может служить заменой мыши или клавиатуры. На основе подобного контроллера ATmega16U2 также сделан встроенный адаптер USB-UART в плате Arduino Uno современной версии Rev.3.

ГЛАВА 2

Общее устройство, организация памяти, тактирование, сброс

Изучение Arduino традиционно начинается с краткого ознакомления с функциональностью внешних выводов и возможностями среды программирования. После этого вам предлагают написать традиционную программку мигания светодиодом, указывают на веб-ресурсы по решению типовых задач, а также на справочник по языку, и считается, что первый урок вы усвоили, — можете приступать к работе.

Эта подкупающая простота оборачивается тем, что вы можете многие годы программировать контроллеры Arduino и так и не узнать, как на деле реализуются функции `delay()` или `millis()`, каковы вообще возможности таймеров, для чего их еще можно использовать, и как это делать правильно, не мешая другим функциям. Поэтому мы здесь такого верхоглядства не допустим и выберем более консервативный подход — начнем все-таки с общего обзора структуры AVR-контроллеров, а в следующей главе рассмотрим кратко те их возможности, которые в Arduino обычно вовсе выносятся за скобки. Практические примеры вы начнете осваивать, начиная с *главы 6*, так что пока наберитесь терпения — без этих предварительных сведений вообще не стоило бы затевать всю историю с ассемблером.

Заметки на полях

Ручаюсь, что после прочтения некоторых разделов этой главы у вас в голове образуется полная каша — все нюансы различий в разных моделях контроллеров запомнить нереально (хотя бы потому, что моделей 8-разрядных AVR существует около 200, и все они хоть в чем-то да различаются). Не стоит, однако, впадать в уныние — здесь вам представлена общая картина, так сказать, набросок панорамы с высоты птичьего полета. В дальнейшем (см. *главу 4*) мы ограничимся несколькими моделями контроллеров, для которых детально будет расписано, что и где нужно устанавливать в тех или иных условиях эксплуатации. А руководствуясь общими сведениями из этой главы, вы уже будете представлять, на что обращать внимание при необходимости использовать другие модели контроллеров.

Общая структура внутреннего устройства МК AVR приведена на рис. 2.1. Здесь показаны все основные компоненты AVR (за исключением модуля JTAG, который имеется не во всех моделях, и мы его касаться не будем). В отдельных моделях

обоих семейств некоторые составляющие могут отсутствовать или различаться по характеристикам, неизменным остается только общее 8-разрядное процессорное ядро (GPU, General Processing Unit).

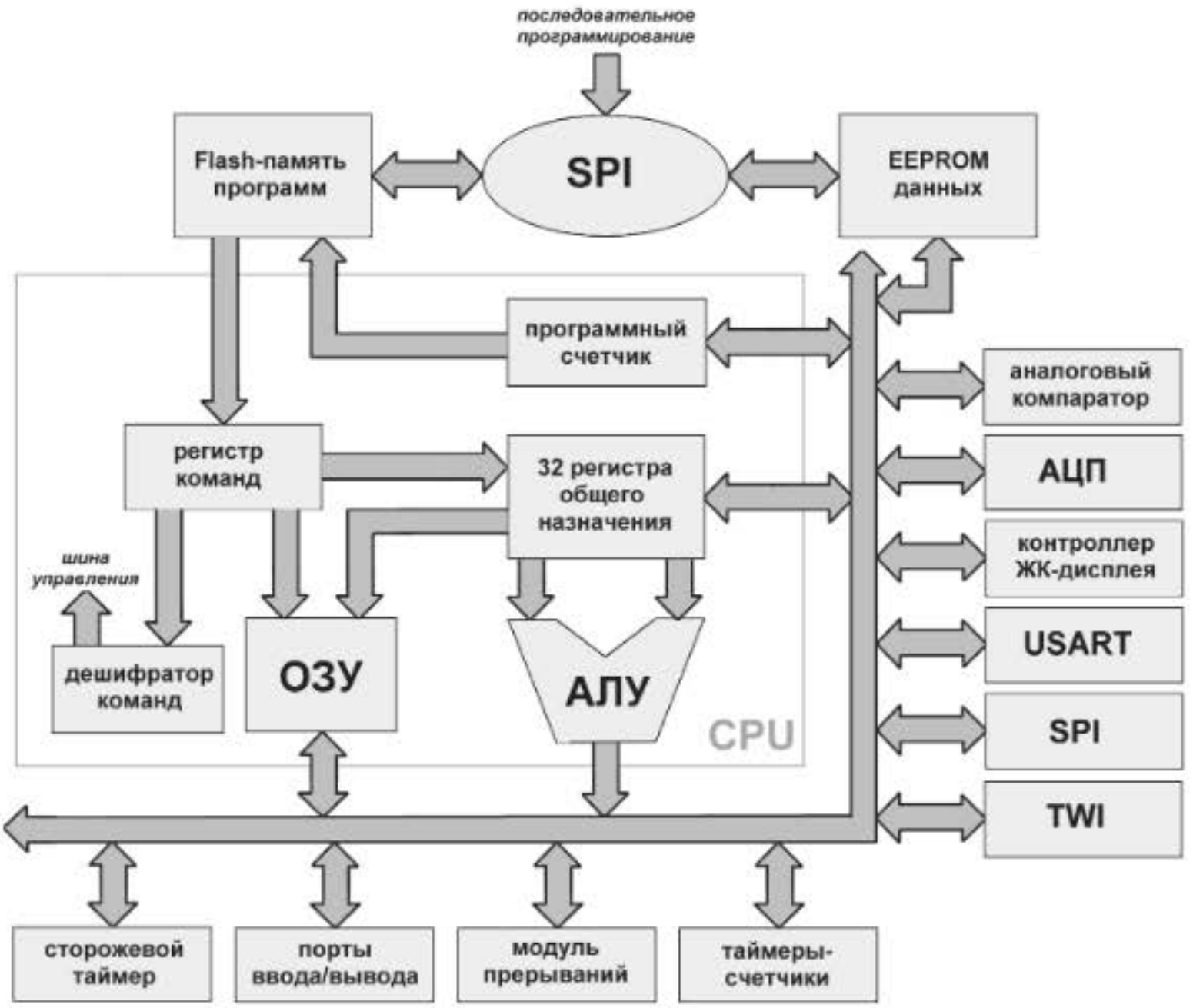

Рис. 2.1. Общая структурная схема микроконтроллеров AVR

Кратко опишем наиболее важные компоненты, большинство из которых мы подробно будем изучать в дальнейшем.

И начнем с памяти. В структуре AVR имеются три разновидности памяти: flash-память программ, ОЗУ (SRAM) для временного хранения данных и энергонезависимая память (EEPROM) для долговременного хранения констант и данных. Рассмотрим их по отдельности.

Память программ

Объем встроенной flash-памяти программ в 8-разрядных AVR-контроллерах составляет от 0,5 кбайта у ATtiny4 до 256 кбайт у ATmega2560. Как мы уже отмечали, первое число в наименовании модели (в серии Tiny — с оговорками для некоторых младших моделей с объемом памяти менее 2 кбайт) соответствует величине этой памяти из ряда: 1, 2, 4, 8, 16, 32, 64, 128 и 256 кбайт. Память программ, как и любая другая flash-память, имеет страничную организацию (размер страницы,

в зависимости от модели, составляет от 64 до 256 байтов). Страница может программироваться только целиком. Число циклов перепрограммирования достигает 10 тысяч.

С точки зрения программиста память программ можно считать построенной из отдельных ячеек — слов по 2 байта каждое. Устройство памяти программ (и только этой памяти!) по двухбайтовым словам — очень важный момент, который нужно твердо усвоить. Такая ее организация обусловлена тем, что бóльшая часть команд в AVR имеет длину ровно 2 байта. Исключение составляют команды `JMP`, `CALL` и некоторые другие (например, `LDS`), которые оперируют с 16-разрядными и более длинными адресами, — длина этих команд равна четырем байтам, и они встречаются лишь в моделях с памятью программ объемом свыше 8 кбайт (подробнее об этом рассказано в *главе 6*). Во всех остальных случаях счетчик команд сдвигается при выполнении очередной команды на 2 байта (одно слово), поэтому необходимую емкость памяти легко подсчитать, зная число используемых ассемблерных команд (вот еще один плюс ассемблера в сравнении с языками высокого уровня). Абсолютные адреса в памяти программ (указываемые, например, в таблицах векторов прерываний в техническом описании МК) также отсчитываются в словах (а не в байтах!), что нужно учитывать при прямом обращении к памяти программ.

Заметки на полях

Приведем пример интересного случая адресации, который представляет команда для чтения констант из памяти LPM (а также ELPM в МК с памятью программ 128 кбайт и более). Эта команда подразумевает чтение по *байтовому* адресу, указанному в двух старших РОН (образующих т. н. *регистр* `Z`, см. об этом далее). Однако, чтобы не нарушать «чистоту» концепции организации памяти программ по словам, разработчики решили этот вопрос следующим образом: в описании команды указано, что старшие 15 разрядов регистра `Z` адресуют *слово* в памяти, а младший разряд выбирает младший или старший байт этого слова (при равенстве разряда 0 или 1 соответственно). Легко, однако, заметить, что байтовая и пословная организации памяти при таком подходе полностью эквивалентны.

Последний адрес существующего объема памяти программ для конкретной модели обозначается константой `FLASHEND`. По умолчанию все контроллеры AVR всегда начинают выполнение программы с адреса `$0000`[1]. Если в программе нет прерываний, то с этого адреса может начинаться прикладная программа. В противном случае по этому адресу располагается т. н. таблица *векторов прерываний*, подробнее о которой мы будем говорить в *главах 3* и *6*. Здесь укажем лишь, что по умолчанию первым в этой таблице (по тому же адресу `$0000`) почти всегда размещается вектор сброса `RESET`, который указывает на процедуру, выполняющуюся при сбросе МК (в том числе и при включении питания).

---

[1] В ассемблере AVR можно обозначать шестнадцатеричные числа в «паскалевском» стиле, предваряя их знаком `$`, при этом стиль языка C (0x00) тоже действителен, а вот «интеловский» способ (00h) не работает. В тексте книги мы будем пользоваться в основном «паскалевским» способом из-за его краткости. Подробнее об обозначениях чисел различных систем счисления в AVR-ассемблере рассказано в *приложении 1*.

В последних адресах памяти программ контроллеров семейства Mega может располагаться т. н. *загрузчик* (Bootloader) — специальная программа, которая управляет загрузкой и выгрузкой прикладных программ из основного объема памяти. В этом случае положение вектора сброса и всей таблицы векторов прерываний (т. е. фактически начального адреса, с которого начинается выполнение программы) может быть изменено установкой специальных конфигурационных ячеек (см. *главу 5*).

Память данных (ОЗУ, SRAM)

В отличие от памяти программ, адресное пространство памяти данных адресуется побайтно (а не пословно). Адресация полностью линейная, без какого-либо деления на страницы, сегменты или банки, как это принято в некоторых других системах. Надо отметить, что в составе семейства Tiny осталась одна модель без памяти данных вообще (ATtiny28L), остальные младшие МК семейства Tiny имеют SRAM объемом от 32 байт. В других моделях объем встроенной SRAM колеблется от 128 байт в представителях семейства Tiny (например, у ATtiny2313) до 4–8 кбайт у старших моделей Mega.

Как видите, ОЗУ контроллера — совсем не то, что ОЗУ компьютерных систем, где его объем измеряется гигабайтами (а если привосокупить к нему объем жестких дисков, которые формально находятся в том же пространстве памяти, то в современных системах уже и терабайтами). Причем львиную часть памяти компьютеров занимают не программы, а именно данные: тексты, картинки, видео. В 8-битных контроллерах, которые для обработки таких объемных данных не предназначены, программы обычно занимают куда больший объем, чем данные.

Адресное пространство статической памяти данных (SRAM) условно делится на несколько областей (рис. 2.2). Темной заливкой выделена часть, относящаяся к собственно встроенной SRAM, до нее по порядку адресов расположено адресное пространство регистров: первые 32 байта занимают регистры общего назначения (РОН), еще 64 — регистры ввода/вывода (РВВ). На рис. 2.2 показан максимальный объем SRAM, равный 8 килобайт (последний адрес `$21FF` — у контроллера ATmega2560, например), в большинстве моделей она меньше, но принцип остается тем же.

Заметки на полях

В архитектуре МК AVR понятие «ввода/вывода» употребляется в двух смыслах: во-первых, имеются «порты ввода/вывода» (I/O ports), которые мы рассмотрим далее. Во-вторых, «регистрами ввода/вывода» (РВВ, I/O registers) в структуре AVR называются регистры, которые обеспечивают доступ к дополнительным компонентам, внешним по отношению к GPU, за исключением ОЗУ (в том числе и к портам ввода/вывода). Такое подразделение приближает структуру МК AVR к привычной конфигурации персонального компьютера, где доступ к любым внешним по отношению к центральному процессору компонентам, кроме памяти, осуществляется через порты ввода/вывода.

У старших моделей Mega все регистры устройств, внешних по отношению к ядру, в 64 адреса не умещаются, поэтому, например, уже у ATmega88 для дополнительных РВВ выделяется отдельное адресное пространство (от `$60` до максимально

возможного в байтовой адресации значения `$FF` — итого таких дополнительных регистров может быть всего 160). Разработчики ядра в свое время не предусмотрели возможности этого расширения, вследствие чего такая модернизация влечет за собой изменение в способах адресации этой расширенной части регистров. Отметим, что в этой книге — чтобы не усложнять программы — мы намеренно не будем использовать моделей контроллеров с расширенным количеством регистров, хотя о способах их адресации, разумеется, расскажем (см. *главу 7*).

Рис. 2.2. Адресное пространство статической памяти данных (SRAM) микроконтроллеров AVR

Для некоторых моделей Mega (ATmega8515, ATmega64, ATmega161, ATmega128, ATmega2560 и др.) предусмотрена возможность подключения внешней памяти SRAM объемом до 64 кбайт (конечный адрес `$FFFF`) с параллельным интерфейсом, так что она становится продолжением внутренней памяти. В настоящее время гораздо удобнее применять для хранения данных внешнюю память с последовательным интерфейсом (аналогично тому, как в компьютерных системах хранят данные на внешних дисках — см. *главу 12*), потому вопросы о подключении внешней параллельной памяти мы обойдем.

Отметим, что адреса РОН и РВВ не отнимают пространство у ОЗУ данных — так, если в конкретной модели МК имеется 512 байт SRAM, а пространство регистров

занимает первые 96 байт (до адреса `$60`), то адреса SRAM займут адресное пространство от `$0060` до `$025F` (т. е. от 96-й до 607-й ячейки включительно, что и составит ровно 512 байт). Конец встроенной памяти данных обозначается константой `RAMEND`, которая для каждой модели контроллера имеет свое значение. Не очень значимое исключение представляет адресация подключаемой внешней памяти упомянутых ранее моделей Mega — т. к. ее максимальный адрес ограничен значением `$FFFF` (65535), то пространство, занимаемое регистрами, вычитается из этой величины.

Операции чтения/записи в память одинаково работают с любыми адресами из доступного пространства, и с этим связано несколько нюансов при употреблении ассемблерных команд. При работе с SRAM нужно быть внимательным — вместо записи в память вы легко можете «попасть» в какой-нибудь регистр. Например, команда загрузки значения регистра `r16` в регистр `r0` (`mov r0,r16`) равносильна записи в SRAM по нулевому адресу (`sts $0000,r16`), т. к. адрес в памяти для РОН совпадает с его номером. Для того чтобы записать что-то в первую ячейку именно SRAM, необходимо обратиться по адресу `$60` (а в контроллерах с дополнительными регистрами — по адресу `$100`). Чтобы не заниматься вычислениями для каждого контроллера, первый адрес памяти SRAM определяется константой `SRAM_START`.

В то же время для непосредственной записи в РВВ по его адресу в памяти к номеру регистра следует прибавить `$20` — так, регистр флагов `SREG`, который для многих моделей располагается в конце таблицы РВВ по адресу `$3F`, в памяти имеет адрес `$5F`. Устанавливать РВВ прямой адресацией памяти необходимо лишь при наличии этих самых дополнительных регистров, а в остальном пользоваться ей неудобно — такая запись всегда отнимает 2 такта вместо одного, характерного для большинства других команд (хотя иногда это позволяет обойти ограничения на манипуляции с некоторыми РОН и РВВ). Именно поэтому мы постараемся здесь не пользоваться контроллерами с расширенным количеством регистров — увлеченному ассемблерщику, возможно, все эти нюансы греют душу, а нормального человека они только запутывают.

В наших программах мы будем обращаться к РОН и РВВ с помощью специально предназначенных для этого команд, где абсолютные адреса скрыты за мнемоническими обозначениями регистров. Однако забывать о наличии всех этих нюансов не следует — так, при переносе готовой программы, работающей с SRAM, на старшие модели контроллеров нужно быть внимательным из-за того, что в них младшие адреса SRAM могут перекрываться дополнительными РВВ.

Энергонезависимая память данных (EEPROM)

Все модели МК AVR (кроме самых младших представителей Tiny) имеют встроенную EEPROM для хранения констант и данных при отключении питания. В разных моделях объем ее варьируется от 64 байт (ATtiny13) до 4 кбайт (старшие модели Mega). Конец EEPROM обозначается константой `EEPROMEND`, причем, если этой памяти нет вовсе, константа будет равна нулю. Число циклов перепрограммирования EEPROM может достигать 100 тысяч.

EEPROM отличается от Flash возможностью выборочного программирования побайтно (вообще-то даже побитно, но здесь этот способ недоступен пользователю, — подробно об этом рассказано в *главе 12*). Для единообразия технологического процесса EEPROM в AVR-контроллерах просто эмулируется за счет части flash-памяти, отсюда некоторые ее особенности в сравнении с «обычной» EEPROM (например, то, что запись в EEPROM не может выполняться одновременно с записью во flash-память). Но на практике, как при загрузке по последовательному каналу (т. е. через SPI-интерфейс программирования), так и при записи или чтении EEPROM из программы, эта особенность не имеет значения, и доступ осуществляется побайтно.

Чтение из EEPROM формально осуществляется в течение одного машинного цикла (правда, на практике оно растягивается на 4 цикла, во время которых CPU просто останавливается, но программисту следить за этим специально не требуется). А вот запись в EEPROM протекает значительно медленнее и к тому же с разной скоростью у разных моделей — цикл стирания/записи может занимать от ~3,4 до ~8,5 мс (в среднем меньшая величина у Tiny и более новых Mega, и большая — у старых Mega, но если эта величина критична, то необходимо ее уточнять для каждой конкретной модели). К тому же процесс записи регулируется встроенным RC-генератором, частота которого нестабильна (при более низком напряжении питания можно ожидать, что время записи будет больше). За такое время при обычных тактовых частотах МК успевает выполнить несколько тысяч команд, так что программирование процедуры записи требует аккуратности — например, нужно следить, чтобы в момент записи не «вклинилось» прерывание (подробнее об этом рассказано в *главах 6* и *8*).

Главная же сложность при работе с EEPROM — возможность повреждения ее содержимого при недостаточно быстром снижении напряжения питания в момент выключения. Обусловлено это тем, что при уменьшении напряжения питания до некоторого порога (ниже порога стабильной работы, но недостаточного для полного выключения) из-за колебаний напряжения МК начинает выполнять произвольные команды, в том числе может осуществить процедуру записи в EEPROM. Если учесть, что типовая команда МК AVR выполняется за десятые доли микросекунды, то ясно, что никакой реальный источник питания не может обеспечить снижение напряжения до нуля за нужное время. В экспериментах автора с семейством Classic при питании от обычного стабилизатора типа LM7805 с рекомендованными значениями емкости конденсаторов на входе и на выходе содержимое EEPROM неизбежно портилось примерно в половине случаев. В современных семействах Mega и Tiny вероятность порчи куда ниже, в основном из-за наличия встроенного монитора питания (Brown-out Detector, BOD — см. о нем далее в этой главе), но тоже не равна нулю, отчего в критичных случаях приходится принимать дополнительные меры (подробнее об этом рассказано далее — в *разд. «Сброс»* этой главы, а также в *главе 10*).

Отметим, что, хотя в документации это не отражено, вероятность порчи содержимого, видимо, резко снижается, если в программе отсутствуют процедуры записи в EEPROM, а константы записывают в нее только при программировании МК.

Бóльшая сохранность данных в таких случаях подтверждается и эмпирическими наблюдениями, и тем, что разрешение записи в EEPROM — процедура двухступенчатая, и случайное возникновение такой последовательности команд практически исключено.

Способы тактирования

За тактирование в основном отвечают конфигурационные ячейки (они же fuse-биты) CKSEL, но их количество и комбинации включения у разных моделей могут различаться очень существенно, потому их всегда нужно проверять по документации (хорошо помогают уже упомянутые ранее пособия Евстифеева [6,7]).

Канонический способ тактирования МК — подключение кварцевого резонатора к соответствующим выводам XTAL1 и XTAL2 (рис. 2.3, *а*). Емкость конденсаторов C1 и C2 в типовом случае должна составлять 15–22 пФ (может быть увеличена до 33–47 пФ с одновременным повышением потребления). Кварцевый резонатор (для краткости его часто называют просто «кварцем») также можно заменить керамическим. Такой возможностью сейчас, вероятно, уже мало кто пользуется — это когда-то керамические резонаторы стоили принципиально дешевле более стабильных кварцевых, и замена имела смысл. Все эти возможности при резонансных частотах кварца от 0,4 МГц до максимальной тактовой частоты контроллера у всех моделей Tiny и Mega реализуются одинаково, и у разных моделей различаются лишь способом задания режима.

Рис. 2.3. Способы тактирования МК AVR с использованием внешних элементов: *а* — кварцевого резонатора; *б* — внешнего генератора; *в* — RC-цепочки

Подробности

У многих контроллеров Mega предусмотрена возможность работы от «часового» кварца (частотой 32,768 кГц), для чего надо соответствующим образом выставить конфигурационные ячейки. Схема ничем не отличается от рис. 2.3, *а*, за исключением того, что рекомендуемая емкость конденсаторов C1 и C2 здесь равна 36 пФ. Если вам зачем-то это понадобится, то проверяйте наличие этой возможности по документации (раздел «даташита» должен называться *Low-frequency Crystal Oscillator*) — например, у ATtiny2313 она отсутствует, и в низкочастотном режиме эта модель может быть

включена только от встроенного или внешнего генератора. Режим работы от «часового» кварца может понадобиться лишь в исключительных случаях — потребление кристалла при этом падает менее, чем до 100 мкА, оказывается удобно отсчитывать промежутки времени с большой точностью, но и множество функций остаются за пределами возможностей (например, при такой частоте невозможно «завести» последовательный порт UART со стандартными значениями скорости обмена). Заметим, что у некоторых моделей (например, у ATtiny2313 — как бы в компенсацию невозможности подключения «часового» кварца) имеется еще отдельный внутренний генератор 128 кГц. В таком режиме потребление тоже падает примерно до 100 мкА, но к нему также относятся все оговорки относительно работы на «часовой» частоте. Более подробно целесообразность эксплуатации контроллеров на сверхнизких частотах мы рассмотрим в *разд. «Потребление МК AVR» главы 4*.

В большинстве моделей Mega имеется специальный конфигурационный бит CKOPT, который позволяет регулировать потребление. При установке этого бита в 1 (незапрограммированное состояние) размах колебаний генератора уменьшается, однако при этом сужается возможный диапазон частот и снижается общая помехоустойчивость, поэтому при высоких частотах рекомендуется сменить этот режим на полный размах (CKOPT = 0).

Автору этих строк удавалось запускать МК на нестандартных частотах, используя вместо кварца в том же подключении миниатюрную индуктивность (при ее значении 4,7 мкГн и емкостях конденсаторов 91 пФ частота получается около 10 МГц).

Заметки на полях

При тактировании от кварцевого резонатора возникает законное желание получить максимальную точность отсчета промежутков времени. При этом надо учитывать, что кварцевые резонаторы бывают очень разные. Погрешность порядка 10^{-4}, которую почти гарантированно можно получить от любого кварца, кажется очень маленькой, и для огромного большинства применений это так и есть — ни одну физическую величину, кроме времени, вы в домашних условиях с такой точностью (0,01%) не измерите. Но сама по себе эта величина погрешности не так уж и хороша — часы с такой точностью будут уходить на минуты в месяц, что уже почти неприемлемо. Так что при необходимости получить более высокую точность смотрите, что приобретаете, — при прочих равных можно ожидать, что кварц в полноразмерном корпусе HC-49U будет точнее и стабильнее, чем в обычном низком «гробике» HC-49S (проверить это можно по документации на конкретную марку). Кроме того, стабильность зависит от схемы генератора, над которой вы тут не властны. Поэтому, чтобы гарантированно получить точный отсчет времени, лучше употреблять не простые кварцевые резонаторы, а готовые прецизионные кварцевые генераторы, которые гарантируют точность и стабильность частоты (самые популярные выпускает фирма Epson).

Вторая возможность тактирования, наличествующая у всех без исключения современных Tiny и Mega, — от встроенного RC-генератора (Internal RC Oscillator). Она не требует внешних элементов, потому включена по умолчанию. Для работы от внешнего кварца следует не забывать перестроить конфигурацию fuse-битов, иначе вы можете быть весьма удивлены результатами работы, например, таймеров. А вот возможные частоты этого встроенного генератора могут отличаться у разных моделей. У «наших» моделей Mega (т. е. тех, на которые мы будем преимущественно ориентироваться в этой книге) — можно выбирать из ряда: 1, 2, 4 и 8 МГц (1 МГц по умолчанию), у Tiny2313 — из двух значений: 4 и 8 МГц (8 МГц по умолчанию).

А в других случаях значения могут быть сильно различаться — например, у «любимого» ардуиновского ATmega328 (как и его аналогов с меньшим объемом памяти Mega88 и Mega168) доступна только одна встроенная частота 8 МГц, зато есть некоторые модели (весьма редкие), у которых встроенный генератор может работать на частоте 20 МГц и на «часовой» частоте 32,768 кГц.

Обратите внимание, что у многих контроллеров при тактировании от встроенного генератора выводы XTAL1 и XTAL2 могут использоваться как дополнительные выводы обычных портов. У ATtiny2313, например, это два вывода в остальном отсутствующего порта A (PA0 и PA1), у ATmega8 — дополнительные выводы порта B (PB6 и PB7) и т. д. Особенно выгодно это использовать у младших Tiny с малым количеством контактов корпуса.

На практике следует ожидать, что у всех моделей, вне зависимости от частоты генератора, *тактовая частота «чистого» кристалла по умолчанию будет равна 1 МГц*. Дело в том, что даже у тех контроллеров, у которых частота встроенного генератора не может быть установлена равной 1 МГц (ATtiny2313, ATmega328 и т. п.), имеется конфигурационная ячейка под названием `CKDIV8`, которая по умолчанию запрограммирована (установлена в нулевое значение), что означает деление частоты генератора на 8. Не забывайте ее переустановить перед использованием внешнего резонатора (отметим, что в русскоязычных пособиях Евстифеева [6,7] этот момент отражен нечетко, так что всегда смотрите документацию на конкретный контроллер!). У некоторых моделей (ATtiny2313) имеется также внутренний делитель тактовой частоты (регистр `CLKPR`), с помощью которого частоту тактирования можно менять программно. По идее эта возможность работает при любом способе тактирования (так можно, например, регулировать потребление контроллера), но более актуальна она при тактировании от встроенного генератора, частоту которого саму по себе мы изменить не можем.

Частоту встроенного генератора, которая и сама по себе имеет разброс от одного кристалла к другому и «плавает» от изменения внешних условий (температуры и напряжения питания), можно регулировать весьма в широких пределах. Но пользоваться возможностями такой калибровки нужно очень осмотрительно — документация не рекомендует менять частоту более, чем на 10% от номинальной, иначе могут быть сложности, например, с записью во flash- и EEPROM-память. Для обычных применений заводской калибровки вполне достаточно, и усложнять себе жизнь, пытаясь стабилизировать параметры заведомо нестабильной схемы, не имеет смысла — проще подключить стабильный внешний кварц или, если очень надо, готовый прецизионный генератор.

Заметки на полях

В одной из публикаций на сайте, посвященном конструированию всяческих автомобильных прибамбасов, мне встретились жалоба на отказы встроенного генератора Tiny2313 при температурах ниже примерно –20 °C и указание на то, что при этом помогает установка обычного внешнего кварца. В документации на контроллеры про это ничего не сказано, наоборот, приведен график изменения частоты RC-генератора от температуры вплоть до граничного рабочего значения минус 40 °C (причем при снижении температуры частота у некоторых моделей повышается, у других — снижается, но эти изменения не очень существенные, на 2–3% от среднего значения). Потому

есть подозрение, что контроллер тут ни при чем, но в домашних условиях досконально проверить это нереально — нужны лабораторные климатические испытания при низких температурах. Можно только тщательно проанализировать схему на предмет работоспособности всех компонентов при таких температурах и проверить надежность паек, которые при температурной деформации могут отходить. И все-таки стоит держать возможность отказов в уме и по возможности проводить натурные испытания — не во всех реальных случаях можно теоретически учесть все возможные сочетания причин.

Все модели AVR также имеют возможность тактирования от внешнего генератора (рис. 2.3, *б*), ограниченного только значением максимальной частоты самого контроллера. Сигнал, разумеется, должен соответствовать требованиям к уровням и таймингам сигналов, установленных для контроллера. На практике эта возможность может быть востребована при необходимости строго синхронизировать работу нескольких контроллеров, а также для получения строго определенной тактовой частоты от внешнего прецизионного генератора.

Отметим, что случайная установка в режим работы от внешнего генератора (этот режим у всех AVR задается одинаково — все имеющиеся ячейки `CKSEL` переводятся в нулевое значение) делает невозможным внутрисхемное программирование контроллера. Эта распространенная ошибка новичков — следствие запутанной терминологии в установках конфигурационных ячеек и, как следствие, разнобоя их установок в разных программаторах. Подробнее этот вопрос мы обсудим в *главах 4* и *5*, а здесь заметим, что паника в этом случае совершенно необоснованна, — если кристалл перестал откликаться, соберите любой внешний генератор из деталей, которые есть под рукой (подробнее об этом рассказано в *главе 5*). А вообще-то все равно следует обзавестись готовым цифровым генератором (о чем речь пойдет в *главе 4*) — он и в работе с Arduino требуется очень часто.

Еще одна возможность тактирования связана с использованием внешней RC-цепочки, подключаемой к выводу XTAL1 (рис. 2.3, *в*). Заметим, что этот способ доступен не у всех моделей (в частности, у ATtiny2313 такая возможность отсутствует). Он задействуется в случаях, когда стабильность частоты не важна, а возможности встроенного генератора по какой-либо причине вас не устраивают. Частота при таком подключении рассчитывается по примерной формуле $f \approx 1/3RC$. Из ограничений на элементы R и C (R в диапазоне 3,3–100 кОм, минимальное значение C = 22 пФ) вытекает, что максимальная частота, достижимая таким способом, равна примерно 5 МГц.

Заметки на полях

Интересное применение этого способа тактирования МК — динамическая подстройка частоты в системах измерения, регулирования и контроля. Например, когда ваш МК задействован в системе управления синхронным двигателем или генератором, работающим с частотой бытовой сети, может потребоваться точная синхронизация управляющих сигналов с сетевой частотой 50 Гц. В этом случае несложно заменить сопротивление R на регулируемый извне преобразователь «частота — значение тока» и получить простую в налаживании аналоговую автоподстройку вместо долгой возни с алгоритмами цифрового регулирования частоты и фазы. Разумеется, при точно не известной и тем более переменной тактовой частоте некоторые функции МК будут недоступны (передача по UART, например), но в таких ситуациях они, как правило, и не требуются.

Отметим еще для галочки, что в некоторых моделях (ATmega8, ATmega8535 и др.) имеются внутренние конденсаторы 36 пФ, которые можно подключить, если записать логический ноль в конфигурационную ячейку `CKPOT`. Они оказываются подключенными между выводами XTAL1 и XTAL2 и «землей», подобно конденсаторам на схеме рис. 2.3, *а*. Иными словами, их можно использовать в схемах с низкочастотным кварцем или в схеме с внешней RC-цепочкой, а вот в основной схеме с кварцевым резонатором их лучше не употреблять, т. к. номинал их почти вдвое выше рекомендуемого (от этого может заметно возрасти потребление и генератор будет перегружаться). Способ этот не универсальный (во многих моделях контроллеров эти конденсаторы отсутствуют), да и сами способы тактирования, где он пригоден, достаточно экзотичные, поэтому мы его употреблять не будем.

Неправильным будет также не упомянуть, что во многих моделях (из «наших» это ATtiny2313) имеется предделитель тактовой частоты, управляемый программно битами `CLKPS` (не путать с fuse-битами `CKDIV8` и `CKSEL`, управляющими собственно значением частот генератора и устанавливаемыми до загрузки программы). В ATtiny2313 можно поделить частоту генератора по всему двоичному ряду от 1 до 256. Такое может понадобиться, например, если от внутреннего генератора, который обычно сам по себе работает на частоте 8 МГц, захочется получить некую частоту тактирования, отличную от «умолчательной» 1 МГц. Возможно, у кого-то возникнет искушение попробовать работу на суперНизких частотах — при исходных 8 мегагерцах деление на 256 позволяет получить тактовую частоту меньше 32 кГц. Для КМОП-схем это означает практическое выключение: потреблять будет только сам генератор, что, впрочем, может быть тоже немало — при 5 вольтах питания он потребляет около 50 мкА.

Сброс

Сбросом (RESET) называется установка начального режима работы МК (аппаратная или, как это раньше называли, «холодная» перезагрузка). При этом все РВВ устанавливаются в состояние по умолчанию — как правило, это нули во всех разрядах за небольшим исключением. На практике то же самое относится и к РОН (как и к ячейкам SRAM) — они по сбросу обнуляются, но не случайно этот момент никак не оговорен официально. В некоторых случаях (например, после выхода из «сна» — об этом см. *главу 14*) РОН и ячейки SRAM после сброса могут принимать произвольные значения, поэтому при необходимости обязательно начинать с какой-то определенной величины переменные следует устанавливать в начале программы принудительно.

Программа после аппаратного сброса начинает выполняться не сразу, а с некоторой задержкой (см. описание fuse-битов `SUT0` и `SUT1` в *главе 5*). Выполнение происходит с заданного начального адреса (по умолчанию это адрес `$0000`), что позволяет при желании организовать софтверную («горячую») перезагрузку изнутри программы, просто организовав безусловный переход на нулевой адрес. При этом все регистры и память сохраняют свое состояние на момент сброса. Правда, зачем такое может понадобиться, я так и не придумал — сторожевой таймер в этом качестве задействовать куда удобнее.

А вот при наличии программы-загрузчика контроллер обращается сначала к нему, и запуск откладывается на время ожидания, — а вдруг извне придет команда на перезапись программы? Подробно об этом рассказано в *главе 6*, а здесь только отметим, что время ожидания может составлять несколько секунд, что иногда бывает критично, — еще одно ограничение, присущее в том числе и платформе Arduino.

Сброс всегда происходит при включении питания. Кроме этого, источниками сброса могут быть следующие события:

- аппаратный сброс, т. е. подача низкого уровня напряжения на вход RESET. Поскольку активный уровень сброса у всех микросхем всегда низкий (исключения мне неизвестны), то правильнее его обозначать с инверсией: $\overline{\text{RESET}}$. Если вы встретили такое обозначение, то речь идет именно о выводе сброса, просто функция сброса, понятно, именуется без всякой инверсии;
- — окончание отсчета установленного интервала сторожевого таймера;
- — срабатывание схемы BOD.

Подробности

Значение четырех младших битов регистра состояния `MCUCSR` должно сигнализировать о том, от какого источника производился сброс предыдущий раз (установка в 1 бита `0` — сброс при включении, бита 1 — аппаратный сброс, бита 2 — от схемы BOD, бита 3 — от сторожевого таймера). На практике, по опыту автора, по состояниям этого регистра надежно отличаются от всех остальных лишь состояния сброса по сторожевому таймеру (прочие флаги могут оказаться установленными все одновременно). Тем не менее, эта информация может быть полезной, например, при анализе причин перерывов в работе круглосуточно работающих устройств (см. *главу 14*).

В младших МК семейства Tiny (кроме ATtiny28L) нет встроенного «подтягивающего» резистора на выводе $\overline{\text{RESET}}$, поэтому для надежной работы следует предусмотреть подключение внешнего резистора величиной 2–10 кОм от этого вывода к напряжению питания. Автор также настоятельно рекомендует устанавливать подобный резистор для любых моделей AVR, т. к. встроенный резистор имеет часто большой номинал (30–60 кОм), и на нем могут наводиться помехи, способные привести к непредсказуемому сбросу.

Не помешает также (хотя в технических описаниях такой рекомендации и не содержится) установка конденсатора 0,1–1 мкФ от вывода $\overline{\text{RESET}}$ на «землю» — это сглаживает неизбежный дребезг напряжения и при включении немного затягивает фронт нарастания напряжения на выводе $\overline{\text{RESET}}$ по сравнению с увеличением напряжения питания — когда наступит порог срабатывания схемы сброса, напряжение питания всего МК уже установится. В пользу этой рекомендации говорит также факт, что на всех платах Arduino такая внешняя цепочка 10 кОм/0,1 мкФ по выводу $\overline{\text{RESET}}$ обязательно имеется совместно с кнопкой ручного сброса. Впрочем, если питание организовано надлежащим образом, то нарушение этой рекомендации не приводит к каким-то фатальным последствиям. Конденсатор можно не устанавливать также, если за сбросом следит внешний монитор питания (супервизор — подробнее о нем рассказано далее).