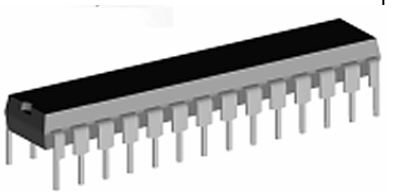

ГЛАВА 10

Аналоговый компаратор и АЦП

В AVR довольно много встроенных возможностей для выполнения операций с аналоговыми величинами: это аналоговый компаратор, который неизменно входит во все без исключения модели AVR (а в "продвинутом" семействе XMega их даже несколько) и 10-разрядный многоканальный АЦП (в семействе XMega он стал 12-разрядным). Преобразования в обратную сторону — цифрового значения в аналоговое — до сих пор можно было осуществлять только с помощью ШИМ-режима таймеров (см. *главу 8*), лишь в семействе XMega появились "настоящие" ЦАП. Справедливости ради заметим, что на практике задача цифроаналогового преобразования возникает значительно реже обратной.

Далее мы разберем аналого-цифровые преобразования с помощью аналогового компаратора и АЦП. Но сначала познакомимся с принципом аналоговых операций, "подводными камнями", которые могут нас поджидать на этом пути, а также основной терминологией по этому вопросу.

Аналого-цифровые операции и их погрешности

Основной принцип оцифровки любых сигналов очень прост (рис. 10.1, *а*). В некоторые моменты времени t_1, t_2, t_3 мы берем мгновенное значение аналогового сигнала и как бы "прикладываем" к нему некоторую меру, линейку, проградуированную в двоичном масштабе. Обычная линейка у нас содержит крупные деления (метры), поделенные на десять частей (дециметры), каждая из которых также поделена на десять частей (сантиметры), и т. д. Двоичная линейка содержала бы деления, поделенные пополам, затем еще раз пополам и т. д. (насколько хватит разрешающей способности). Если вся длина такой

линейки составляет, допустим, 2,56 м, а самое мелкое деление 1 см (т. е. мы можем измерить длину с точностью не хуже 1 см, точнее, даже половины его), то таких делений будет ровно 256 и их можно представить двоичным числом размером 1 байт, или 8 двоичных разрядов. Принцип не изменится, если мы измеряем не длину, а напряжение, ток или сопротивление, только смысл понятия "линейка" будет каждый раз иной.

Так мы получаем последовательные отсчеты величины сигнала x_1, x_2, x_3. Причем заметьте, что при выбранной разрешающей способности и числе разрядов мы можем померить аналоговую величину не больше некоторого значения, которое соответствует выбранному масштабу. Иначе придется или увеличивать число разрядов (длину линейки), или менять разрешающую способность в сторону ухудшения (растягивать линейку). Все изложенное и есть сущность работы аналого-цифрового преобразователя (АЦП).

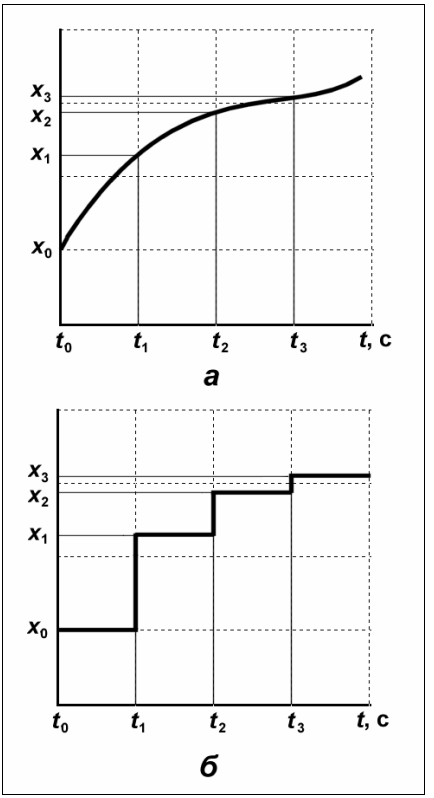

Рис. 10.1. Операции с аналоговыми сигналами:
а — основной принцип АЦП; *б* — обратная операция, ЦАП

Если мы оцифровываем какую-то меняющуюся во времени величину, например звуковой сигнал, то приходится производить измерения регулярно. В этом случае можно говорить о временно́м разрешении преобразования с определенным *битрейтом*, и об искажениях частотного спектра сигнала. С помощью встроенных средств AVR подобные преобразования осуществимы лишь для относительно медленно меняющихся сигналов (для качественной оцифровки звука АЦП и аналоговый компаратор в AVR недостаточно скоростные), потому мы остановимся лишь на задаче, когда требуется преобразовывать статический сигнал при измерении аналоговой величины (т. е. когда измерения проводятся достаточно редко и сама по себе их регулярность роли не играет).

Обратная задача (цифроаналоговое преобразование) проиллюстрирована на рис. 10.1, *б*. Сущность ЦАП в том, что мы выражаем двоичное число в виде пропорциональной величины напряжения, т. е. занимаемся, с точки зрения теории, всего лишь преобразованием масштабов или, иначе, физическим моделированием абстрактной величины — числа. Вся аналоговая шкала поделена на кванты — градации, соответствующие разрешающей способности нашей двоичной "линейки". Как мы видим из рис. 10.1, *б*, второй график представляет первый, мягко говоря, весьма приблизительно. Чтобы повысить степень достоверности полученной кривой, следует увеличить разрядность преобразования. Тогда ступеньки будут все меньше и меньше, и есть надежда, что при некотором достаточно высоком разрешении кривая станет, в конце концов, неотличимой от исходной непрерывной аналоговой линии.

Ясно, что сколь угодно увеличивать разрешение нельзя — число разрядов АЦП ограничено. Поэтому в любом аналого-цифровом преобразовании обязательно присутствует погрешность квантования, связанная с разрядностью АЦП. Но это не единственная составляющая общей погрешности преобразования. Кроме нее есть случайная погрешность (абсолютная погрешность по всей шкале, которая для встроенного АЦП МК AVR может составлять, согласно документации, до ±2 LSB), погрешность нелинейности шкалы (в АПЦ она достаточно мала: ±0,5 LSB) и дополнительная случайная погрешность, связанная с наводками и шумами.

Для борьбы с последней принимают специальные меры. На сигнальном входе следует фильтровать сигнал устанавливая небольшой конденсатор; если аналоговая часть МК питается от того же источника, что и цифровая, то аналоговое питание следует также фильтровать установкой *LC*- или *RC*-фильтров. Не рекомендуется использовать свободные выводы того же порта, к которому подсоединен АЦП (обычно это PortA), для обработки цифровых сигналов во время преобразования. Кроме того, имеется возможность вообще остановить

МК на время преобразования через режим ADC Noise Reduction, о котором далее.

Заметки на полях

Не следует забывать и о том, что для более полной реализации возможностей имеющегося АЦП нужно стремиться к тому, чтобы пределы изменения измеряемой величины были как можно ближе к диапазону АЦП — длине нашей двоичной линейки. Причем именно диапазон изменения, который несет информацию, а не полный размах аналогового сигнала: если сигнал изменяется, к примеру, от 4 до 4,5 В, и мы его будем измерять с помощью АЦП с диапазоном от 0 до 5 В и разрешением 10 двоичных разрядов (что соответствует минимальному делению линейки в 5 В/1024 ≈ 5 мВ), то мы ухудшим результат с точки зрения разрешения ровно в 10 раз: на 0,5 В реального диапазона изменения измеряемой величины придется лишь 100 градаций. А неизменную "подставку", соответствующую величине 4 В, ниже которой исходная величина никогда не опускается, в процессе расчета физического значения всегда можно просто прибавить, получив таким образом результат с наибольшим возможным разрешением. Чаще всего эта величина значения не имеет: например, медный термометр сопротивления всегда имеет определенное значение сопротивления при минимальной температуре диапазона, и оно в общем случае не несет никакой информации. Однако для того, чтобы такую подгонку диапазонов осуществить, нередко приходится идти на заметное усложнение предварительной аналоговой схемы, да и не всегда задача получения именно наивысшего разрешения стоит так уж остро. Примером может служить рассматриваемый далее датчик атмосферного давления, диапазон которого в мм рт. ст. как раз примерно соответствует десятибитовой шкале, даже с некоторым запасом, поэтому вычитать "подставку" в 600 или 700 мм рт. ст., ниже которой давление в данной местности не опускается, оказывается нецелесообразно.

На практике для борьбы со случайными флуктуациями результата измерений, которые, как показывает опыт, "вылезут" обязательно, не взирая ни на какие принятые меры, можно применить еще один прием — усреднение нескольких измерений, и о нем мы также поговорим далее.

До сих пор мы говорили в основном о встроенном АЦП. При необходимости простейший АЦП можно построить на основе аналогового компаратора. Давайте рассмотрим типовые применения компаратора, а потом уже опять вернемся к АЦП более подробно.

Работа с аналоговым компаратором

Аналоговый компаратор сравнивает две величины напряжения, установленные на его входах, и в зависимости от их соотношения устанавливает выход в одно из двух логических состояний. Входы обычно маркируются знаками "плюс" и "минус" (и называются положительным и отрицательным, или, иначе, неинвертирующим и инвертирующим). Если напряжение на положитель-

ном входе превышает напряжение на отрицательном, то выход компаратора устанавливается в логическую единицу, и наоборот — если напряжение на отрицательном входе больше, чем на положительном, то выход устанавливается в логический ноль. В AVR эти входы называются AIN0 (положительный) и AIN1 (отрицательный). На схемах я в дальнейшем для ясности обозначаю их AIN+ и AIN–.

Ошибка аналогового компаратора AVR (напряжение смещения) — не более 40 мВ, время отклика — не более 0,5 мкс. Напряжения на входах не должны выходить за пределы напряжения питания. Аналоговый компаратор может также работать совместно с АЦП, если оно имеется в данной модели МК — инвертирующий вход может подключаться к одному из входов АЦП. Подробности приведены в техдокументации, а здесь отметим, что эта функция может пригодиться, например, для определения знака какой-то физической величины или для сигнализации о ее выходе за допустимые пределы. Еще одну интересную возможность предоставляет функция подключения компаратора ко входу захвата Timer1, что позволяет с его помощью построить формирователь входных импульсов для измерения длительности временных интервалов или для подсчета событий (см. *главу 8*).

Во всех моделях МК AVR компаратор управляется одинаково, через единственный регистр ACSR. Бит 7 (ACD) этого регистра управляет включением компаратора, причем нулевое его состояние (по умолчанию) означает, что компаратор *включен*. Поэтому для энергосберегающих режимов его нужно специально выключать.

Подробности

Положительный (неинвертирующий) вход компаратора путем установки бита ACBG может подключаться ко внутреннему источнику опорного напряжения величиной 1,22 В, тогда напряжения на отрицательном входе будут сравниваться с этой величиной, что позволяет упростить внешнюю схему. Недостаток такого приема состоит в том, что источник этот довольно неточный — разброс его значения может достигать ±0,1 В (около 8%), причем насколько это обусловлено отклонениями в процессе изготовления (что в принципе терпимо, т. к. позволяет индивидуально калибровать схему), а насколько — дрейфом в процессе эксплуатации, из документации понять невозможно. Между тем, такая погрешность зачастую недопустима — даже в рассматриваемой далее простейшей задаче слежения за напряжением резервной батарейки 4,5 В, 8% ошибки дадут разброс срабатывания компаратора более чем на 0,3 В, и в результате может случиться, что схема сработает либо тогда, когда батарейка уже неработоспособна, либо когда в ней еще имеется достаточный запас энергии. Тем не менее, в некритичных к таким ошибкам задачах внутренний источник подойдет, причем вполне правомерно допущение, что при установившейся температуре корпуса МК дрейф источника будет невелик. При подключении неинвертирующего вхо-

да компаратора к внутреннему источнику вывод порта, соответствующий AIN0, можно применять для других целей.

При смене состояния выхода аналоговый компаратор может генерировать прерывание. Для этого нужно задать бит разрешения прерывания ACIE. Два младших бита регистра управления ASIS1:ASIS0 устанавливают событие, вызывающее прерывание: когда оба равны нулю (по умолчанию), то прерывание вызывает любой перепад уровней на выходе компаратора, как из 0 в 1, так и обратно. При этом состояние выхода компаратора в любой момент можно узнать, прочитав бит ACO (бит 5).

Ориентируясь на эти сведения, попробуем решить простейшую задачу слежения за напряжением резервной батарейки, с сигнализацией состояния, когда она требует замены. Будем ориентироваться на батарею 4,5 В, составленную из трех щелочных элементов типа ААА. Признаком неработоспособности батареи считаем падение напряжения ниже 1,1 В на элемент (в сумме 3,3 В). Отметим, что литиевые элементы имеют гораздо более пологую характеристику — обладая начальным напряжением около 3,16 В, практически всю свою емкость они исчерпывают при снижении напряжения примерно на 10% (до напряжения ~2,8 В), после чего очень быстро "сдыхают", и слежение за их состоянием требует более тонкой настройки компаратора.

Контролировать состояние батареи мы будем тогда, когда схема подключена к сетевому источнику — когда МК переключается на батарею, следить за ее своевременной заменой уже поздно. Схема для этого случая приведена на рис. 10.2. Батарея Б1 и внешний источник на стабилизаторе LM78L05 соединены по типовой схеме с развязкой на диодах Шоттки (КД922) с малым собственным падением напряжения (~0,2 В), обеспечивающей автоматическое переключение при исчезновении сетевого питания.

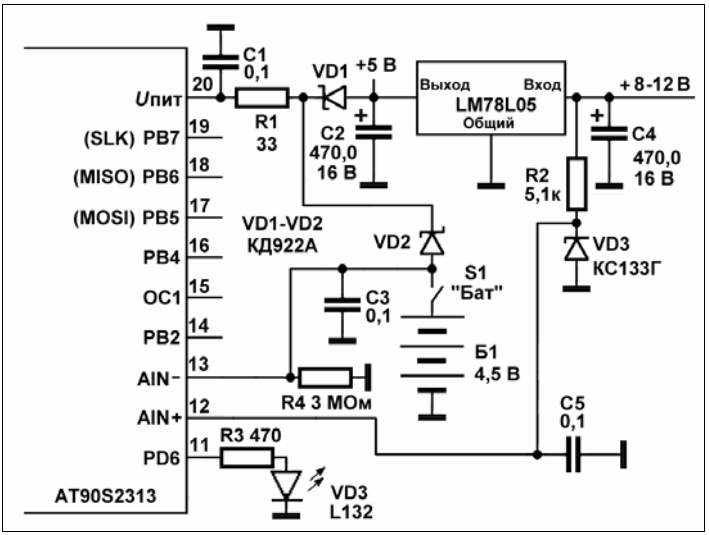

Рис. 10.2. Схема отслеживания напряжения резервной батареи на примере контроллера AT90S2313

Внешним источником опорного напряжения для сравнения служит стабилитрон КС133 с номинальным напряжением 3,3 В. Отметим, стабилитрон имеет собственный разброс, и если напряжение существенно превышает необходимое, то параллельно стабилитрону (и конденсатору C5) можно включить дополнительный делитель.

Пока напряжение батареи, поступающее на вход AIN1 (AIN- на схеме) выше напряжения на стабилитроне (вход AIN+), выход компаратора находится в состоянии логического нуля. Когда напряжение батареи уменьшится ниже 3,3 В, компаратор перебросится в состояние лог. 1 и одновременно возникнет прерывание. В этом прерывании мы включим светодиод, подсоединенный к выходу PB6. Если мы заменим батарею, то компаратор перебросится в обратную сторону и в прерывании мы должны переключить светодиод обратно. Таким образом мы все время будем отслеживать состояние батареи в реальном времени.

Эффект, аналогичный снижению напряжения, даст также отключение тумблера S1 — так можно проверять работу схемы. Резистор R4 сопротивлением 3 МОм нужен, когда батарея отсутствует (или тумблер S1 разомкнут) — иначе вход AIN- повиснет "в воздухе", и компаратор будет срабатывать непредсказуемо. Величина резистора обеспечивает достаточно малый ток (чуть более 1 мкА) и на долговечность батареи практически не влияет.

Наоборот, когда пропадет сетевое питание, на входе AIN+ окажется низкий уровень напряжения, и это равносильно состоянию с исправной батареей,

независимо от ее собственного напряжения. Это позволит не тратить ресурсы батареи на бесполезное свечение светодиода, когда ее все равно менять нельзя, даже если она истощена.

Если все же требуется отслеживать напряжение батареи при отсутствии сетевого питания (например, когда батарея вообще единственный источник), то следует предусмотреть встроенный источник опорного напряжения, как описано ранее, только на входе AIN- параллельно батарее придется установить делитель, чтобы подогнать напряжение срабатывания к уровню 1,22 В. Чтобы не тратить ресурсы батареи зазря, делитель может быть составлен из резисторов с сопротивлением вплоть до единиц мегаом — ток утечки на входе компаратора, который может вносить погрешность в измерения, достаточно мал, и имеет порядок в десятки наноампер.

Часть программы, отвечающая за функцию слежения по такой схеме, очень проста. Сначала следует инициализировать компаратор (т. к. он включен по умолчанию, то специально включать его не нужно, только разрешить прерывание):

```
ldi temp,(1<<ACIE) ;разр. прерывания компаратора при переключении
out ACSR,temp
```

И затем написать обработчик прерывания компаратора (в приведенном на рис. 10.2 МК AT90S2313 это самый последний по счету вектор, назовем его `ACOMPI`), как показано в листинге 10.1.

Листинг 10.1

```
ACOMPI: ;прерывание компаратора
     sbis ACSR,ACO ;если бит ACO =1, то установка LED
     rjmp COMP_0 ;иначе на сброс LED
     sbi PortB,6 ;включаем LED
     reti ;на выход
COMP_0:
     cbi PortB,6 ;гасим LED
reti ;выход из процедуры компаратора
```

Интегрирующий АЦП на компараторе

АЦП простого (однократного) интегрирования на компараторе можно построить, например, по схеме, показанной на рис. 10.3. На основе подобного АЦП построена, например, схема игрового порта IBM PC для оцифровки координат джойстика. Такие АЦП имеют большую погрешность (потому что результат тут зависит от всего подряд: от точности компаратора, дрейфа зна-

чений резистора и емкости, напряжения питания и опорного и т. д.), но зато довольно просты.

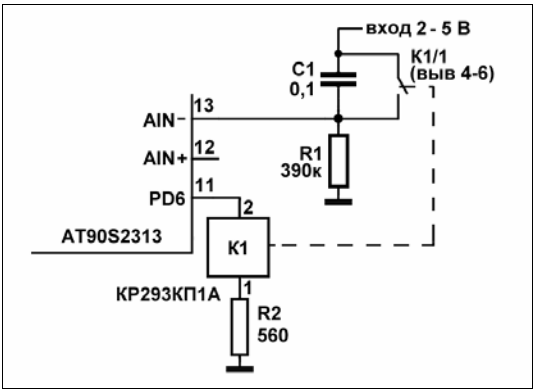

Рис. 10.3. Вариант АЦП однократного интегрирования на встроенном компараторе

Разберем построение программы для этого примера подробно со всеми нюансами, т. к. принципы обращения с данными и перевода их в физические величины здесь точно такие же, как и в других случаях, и нам не придется в дальнейшем отвлекаться на частности. Действие схемы по рис. 10.3 заключается в измерении времени, необходимого для заряда конденсатора после размыкания контактов электронного реле с нуля до порога срабатывания компаратора, который в нашем случае определяется встроенным опорным источником и равен примерно 1,2 В. К сожалению, отсутствие второго (отрицательного) напряжения питания в нашем случае приводит к тому, что шкала входных напряжений ограничена снизу этим порогом срабатывания (а сверху, естественно, напряжением питания). В реальности же ограничение еще более жесткое (на схеме указан нижний порог, равный 2 В), и вот почему.

Принцип работы и расчетные формулы

График изменения напряжения на выводе AIN- в зависимости от времени показан на рис. 10.4. В начальный момент времени контакты электронного реле K1 размыкаются, конденсатор начинает заряжаться, и напряжение на этом выводе падает от начального значения, равного входному, до нуля по экспоненциальному закону, в соответствии с формулой, приведенной на рис. 10.4, внизу. Естественно, ниже порога компаратора (на графике показан серой пунктирной линией) начальное значение быть не может — иначе компаратор не сработает. Но из-за общей нелинейности процесса зависимость интервала времени между началом заряда и достижением порога (которая и есть результат работы АЦП) от величины входного напряжения также имеет нелиней-

ный характер, и при входных напряжениях, близких к порогу, эта нелинейность слишком велика. Поэтому для достижения приемлемой точности результата мы ограничимся входным диапазоном от 2 В до напряжения питания 5 В.

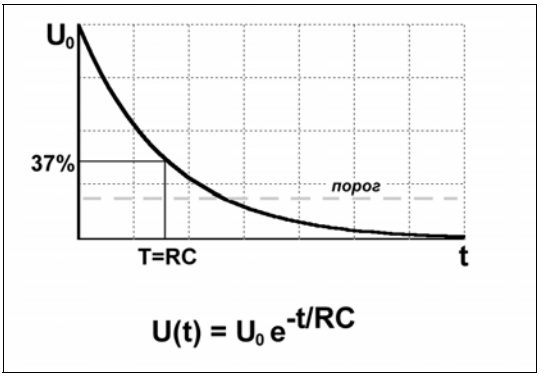

Рис. 10.4. Изменение напряжения на выводе AIN- в процессе заряда конденсатора C1

Время достижения порога, естественно, зависит не только от начального напряжения, но и от емкости конденсатора C1 и сопротивления резистора R1. Для упрощения схемы сброса конденсатора я выбрал простейшее электронное реле КР293КП1А (оно известно еще под названием 5П14А), которое имеет, однако, достаточно большое время срабатывания — порядка долей миллисекунды. Чтобы свести к минимуму погрешность из-за этого промежутка времени, когда реле то ли сработало, то ли еще нет, приходится выбрать достаточно большую постоянную времени — для указанных на схеме номиналов она будет равна 39 мс. Отметим, что в реальности номиналы компонентов имеют большой разброс, и последующие расчеты можно положить в основу проекта, но готовую схему придется калибровать индивидуально.

По формуле, приведенной на рис. 10.4, можно подсчитать время, необходимое для достижения порога, в зависимости от входного напряжения. Рассчитанная для ряда значений напряжения от 2 до 5 В величина времени достижения порога 1,2 В приведена в табл. 10.1.

Таблица 10.1. *Время достижения порога в зависимости от начального напряжения*

$U_{нач}$, В	Время, мс
2,0	19,92
2,5	28,62

3,0	35,73
3,5	41,74
4,0	46,95
4,5	51,55
5,0	55,66

Нам для расчета понадобится обратная зависимость — напряжение у нас выступает неизвестной величиной, поэтому из этих данных выведем зависимость напряжения от времени. Аппроксимация полиномом первой степени дает следующую линейную зависимость (время t подставляется в миллисекундах):

$$U_{\text{нач}} = 0{,}147 + 0{,}084t. \qquad (10.1)$$

Ошибка расчета по этой формуле в среднем не превышает нескольких процентов. Максимальная ошибка около 9% будет вблизи нижнего края диапазона, равного 2 В, и если бы мы не ограничились этим значением, то она бы еще значительно выросла. На эту зависимость мы будем ориентироваться — все равно суммарные погрешности такой примитивной схемы не позволят добиться высокой точности, но для перфекционистов я приведу формулу полинома второй степени, которая аппроксимирует зависимость значительно точнее (максимальная ошибка не превысит 0,6%):

$$U_{\text{нач}} = 1{,}524 + 0{,}00323t + 0{,}00106t^2. \qquad (10.2)$$

При изменении постоянной времени коэффициенты будут меняться пропорционально, потому вычислить их при иных значениях R1 и C1 очень просто: достаточно умножить все выражение на дробь, в числителе которой стоит новая постоянная времени R_1C_1, выраженная в миллисекундах, а в знаменателе — значение 39 мс. Однако на самом деле результаты будут зависеть еще и от других параметров схемы, потому реальное устройство все равно потребует калибровки. Отметим, что для повышения стабильности измерений лучше выбирать конденсатор C1 с фторопластовым (тефлоновым) или полиэтилентерефталатным (лавсановым) диэлектриком (серий К72, К73), хотя они и имеют достаточно большие размеры.

На этом теорию закончим и перейдем к программе. В предположении, что измерения производятся непрерывно, нам требуется обеспечить следующую последовательность действий. В первый момент времени контакты реле должны быть замкнуты (для этого на выводе PB6 нужно установить высокий уровень). В момент начала измерения контакты реле размыкаются, и одновременно запускается таймер. По прерыванию при переходе компаратора из нуля в единицу таймер останавливается и фиксируется результат измерений.

Реле замыкается (после чего нужно выдержать паузу порядка миллисекунды для надежного срабатывания реле и полного сброса конденсатора), обнуляется таймер и можно начинать новый цикл измерений. Более одной миллисекунды задержки не требуется — часть этого интервала займет время, необходимое для срабатывания реле, а сам по себе сброс конденсатора произойдет за несколько микросекунд, т. к. сопротивление замкнутых контактов выбранного реле не превышает 5 Ом.

Чтобы "завести" систему, первое измерение придется провести в основной программе, потом можно действовать через прерывания компаратора. Само измерение займет, как мы видим из табл. 10.1, порядка нескольких десятков миллисекунд, и это время можно использовать, например, для вывода результатов на индикацию или во "внешний мир".

Отметим один тонкий нюанс: если на вход напряжение не поступает (или оно меньше порога), то прерываний компаратора происходить не будет, и система "зависнет" в состоянии, когда все время будет демонстрироваться результат последнего измерения, причем, скорее всего, неправильного — велика вероятность, что напряжение на входе будет выключено как раз посередине цикла очередного измерения. Это, разумеется, некорректно: в подобном случае устройство должно демонстрировать нулевое значение. Поэтому придется предусмотреть систему сброса текущего значения, для чего можно, например, задействовать сторожевой таймер. Второй нюанс заключается в том, что напряжения ниже 2 В, но выше порога, дадут нам неверный результат измерения. Просто учесть это в инструкции по эксплуатации было бы слишком некрасивым решением, потому мы ограничим расчетный временной интервал значением, например, 18 мс, что соответствует по табл. 10.1 напряжению немного ниже 2 В. Видите, как даже самая простая программа обрастает всякими "наворотами", когда дело доходит до практического применения!

Так как высокого разрешения тут не требуется, то для того, чтобы измерять время, возьмем 8-разрядный Timer0. Если его завести при тактовой частоте МК 4 МГц с коэффициентом деления 1/1024, то он будет считать импульсы с частотой 3906,25 Гц. Тогда в интервал времени 55,66 мс, соответствующий, согласно табл. 10.1, напряжению 5 В, уложится 217 импульсов, так что таймер не переполнится даже при наибольшем значении диапазона. При допустимом минимуме входного напряжения в 2 В число импульсов составит $3906{,}25 \times 0{,}01992 = 78$.

Прежде чем приступать к программе, давайте посмотрим, как можно пересчитать полученное значение времени в напряжение, чтобы сразу получить физические величины. Интервал времени, выраженный в миллисекундах, при частоте 3906,25 Гц на входе таймера вычисляется по формуле: $t = n/3{,}90625$,

где n — сосчитанное число импульсов. С учетом этого наша полуэмпирическая формула (10.1) перепишется так:

$$U_{\text{нач}} = 0{,}147 + 0{,}0215n. \qquad (10.1a)$$

Для расчета по этой формуле придется вспомнить материал *главы 7*, касающийся операций с дробными числами. Переведем коэффициенты в область целых чисел, используя коэффициент, кратный степени двойки. Для того чтобы не потерять значащие разряды, придется умножить как минимум на число $2^{14} = 16\,384$, но мы не будем экономить (все равно расчеты придется выполнять как минимум с трехбайтовыми величинами), и возьмем для удобства двухбайтовое значение $2^{16} = 65\,536$. Тогда расчетная формула станет такой:

$$65536U_{\text{нач}} = 9634 + 1409n. \qquad (10.1б)$$

Операции с такими числами мы проводить умеем, а результат потом преобразовать очень просто: достаточно у полученного трехбайтового числа (за его пределы результат не выйдет) отбросить младшие два байта. Однако не забудем, что результат получится в целых вольтах и потому слишком грубый. Чтобы ввести десятые вольта, придется отбросить младший байт, затем умножить полученное число на 10, и только потом отбрасывать еще один байт — при такой операции у нас значения не выйдут за пределы двухбайтовых чисел. Мы могли бы ввести коэффициент 10 прямо в формулу, но тогда значения перемножаемых чисел изначально вышли бы за пределы двух байтов и умножение значительно бы усложнилось.

Вот только у выбранного нами AT90S2313 (как и у его старшего собрата ATtiny2313) инструкции аппаратного умножения отсутствуют, потому придется применить программное умножение. Следующая процедура перемножения двух 16-разрядных чисел с получением 24-разрядного числа (листинг 10.2) получена модификацией процедуры из "аппноты" 200 (с частичным сохранением приведенных там комментариев и учетом допущенной в ней ошибки).

Листинг 10.2

```
Mp16x16x24: ;перемножение двух 16-разрядных величин,
;результат 3 байта, 24 разряда
;dataL multiplicand low byte
;dataH multiplicand high byte
;KoeffL multiplier low byte
;KoeffH multiplier high byte
;temp result byte 0 (LSB)
;temp1 result byte 1
```

```
;temp2 result byte 2 (MSB)
;countCyk loop counter
     clr temp2
     ldi countCyk,16 ;init loop counter
m16u_1: lsr KoeffH
     ror KoeffL
     brcc noad8 ;if bit 0 of multiplier set
     add temp,dataL ;add multiplicand Low to byte 2 of res
     adc temp1,dataH ;add multiplicand high to byte 3 of res
noad8:     ror temp2 ;shift right result byte 2
     ror temp1 ;rotate right result byte 1 and multiplier High
     ror temp ;rotate result byte 0 and multiplier Low
     dec countCyk ;decrement loop counter
     brne m16u_1 ;if not done, loop more
ret
```

Для МК семейства Mega удобнее, конечно, применить процедуру с аппаратным умножением, приведенную в *главе 7*. Заметим, что старший разряд данных (`dataH`) у нас здесь всегда будет равен нулю, но в целях унификации процедуры я не стал ее упрощать — данный пример расчета может использоваться и в других случаях.

После этой основательной подготовки мы, наконец, можем перейти к собственно программе.

Программа интегрирующего АЦП

Листинг 10.3 содержит константы и переменные, которые понадобятся в программе.

Листинг 10.3

```
.equ AdrData = $60
.equ kA0 = 9634 ;свободный член
.equ kA1 = 1409 ;крутизна характеристики
.def dataL = r16
.def dataH = r17
.def temp = r18
.def temp1 = r19
.def temp2 = r20
.def countCyk = r21
.def KoeffL = r22
.def KoeffH = r23
```

Измеренную величину мы будем писать в SRAM по адресу `AdrData` в две последовательные ячейки $60:61 (два десятичных разряда в распакованном BCD), поэтому сначала эти ячейки придется обнулить (в отличие от регистров, ячейки SRAM по сбросу устанавливаются в произвольное состояние). Если измерений долго не будет, МК сбросится сторожевым таймером, и значение обнулится. Из SRAM измеренную величину можно извлекать для индикации, для отправки во "внешний мир" или еще для каких-либо целей — эту часть программы мы оставим "за кадром". В программе далее имеется процедура инициализация сторожевого таймера — обратите внимание, что она годится для МК семейства Tiny или Classic (в частности, для AT90S2313), в контроллерах Mega WD-таймер запускается иначе (см. *главу 14*).

Основную программу, после всех необходимых установок, начнем с инициализации аналогового компаратора (до этого не должно идти команды разрешения прерываний!), как показано в листинге 10.4.

Листинг 10.4

```
. . . . .
ldi temp,(1<<ACIE)|(1<<ACBG)|(1<<ACIS1)|(1<<ACIS0)
;разр. прерывания компаратора при переходе 0->1
;подключить внутренний источник 1,22В
out ACSR,temp
sbi PortB,6 ;включаем реле
rcall Delay_1mc ;задержка на 1 мс
;=====обнуляем память
clr ZH
ldi ZL,AdrData
clr temp
st Z+,temp ;обнуляем ст. байт данных
st Z,temp ;обнуляем мл. байт данных
;=====запускаем WDT на 500 мс:
wdr ;команда на сброс
ldi temp,(1<<WDP0)|(1<<WDP2)|(1<<WDE)
out WDTCR,temp
sei ;разрешаем прерывания
;=====начальный запуск преобразования
cbi PortB,6 ;выключаем реле
clr temp
out TCNT0,temp ;очистка таймера
ldi temp,0b00000101 ;Timer0 включить 1:1024
out TCCR0,temp ;запускаем таймер
Cykle:
```

```
rjmp cykle
;процедура задержки
Delay_1mc: ;1 миллисекунда
ldi temp1,low(1000)
ldi temp2,high(1000)
C_Delay:
  subi temp1,1
  suci temp2,0
brcc C_Delay
ret
```

Листинг 10.5 иллюстрирует обработчик прерывания компаратора.

Листинг 10.5

```
ACOMPI: ;прерывание при переходе из 0 в 1
  clr temp
  out TCCR0,temp ;останавливаем таймер
  sbi PortB,6 ;обнуляем конденсатор
  in dataL,TCNT0 ;извлекаем данные
     cpi dataL,70 ;70 имп. = 18 мс
     brsh Calc_data ;если больше, к расчету
       ldi ZL,AdrData ;иначе обнуляем значение в памяти
       clr temp
       st Z+,temp ;обнуляем ст. байт данных
       st Z,temp ;обнуляем мл. байт данных
     rjmp New_data ;к новому измерению
Calc_data: ;расчет значения
     clr dataH ;здесь всегда 0
     ldi KoeffL,low(kA1)
     ldi KoeffH,high(kA1)
     rcall Mp16x16x24 ;перемножаем, рез. в temp2:temp1:temp
      ;складываем со свободным членом:
      add temp,low(kA0)
     adc temp1,high(kA0)
     adc temp2,dataH ;старший складываем с нулем
     ;результат в temp2:temp1 умножаем на 10:
     ldi dataL,10
     mov KoeffL,temp1
     mov KoeffH,temp2
     rcall Mp16x16x24 ;рез. в temp1 (temp отбрасываем, temp2=0)
      ;преобразуем в BCD и пишем в память:
      mov temp,temp1
```

```
     rcall bin2bcd8 ;процедуру bin2bcd8 см. главу 7
     ;результат temp1 — старший, temp — младший
     ldi ZL,AdrData ;адрес данных
      st Z+,temp1 ;ст. дес. разряд
      st Z,temp ;мл. дес. разряд
     and temp,0b00001111 ;выделяем младшую тетраду
New_data: ;запуск нового измерения
  rcall Delay_1mc ;задержка на 1 мс
  cbi PortB,6 ;выключаем реле
  clr temp
  out TCNT0,temp ;очистка таймера
  ldi temp,0b00000101 ;Timer0 включить 1:1024
  out TCCR0,temp ;запускаем таймер
reti
```

Этот пример может дать вам представление, как обращаться с аналоговыми схемными решениями с помощью МК и осуществлять расчеты физических величин. Рассчитанное десятичное значение напряжения можно непосредственно послать на индикацию или упаковать и отправить во "внешний мир" с помощью UART (использовав для этого другой таймер). Пример этот, правда, несколько искусственный: городить интегрирующие АЦП самостоятельно, как правило, нецелесообразно — проще задействовать один из каналов встроенного АЦП, благо их предостаточно, а АЦП входят во все МК семейства Mega и в некоторые МК семейства Tiny. К их рассмотрению мы сейчас и перейдем.

Встроенный АЦП

Общие сведения об устройстве АЦП в МК AVR приведены в *главе 3*. Здесь мы поговорим о практических аспектах его применения, но сначала остановимся подробнее на управлении АЦП.

АЦП может функционировать в непрерывном режиме (когда по окончании преобразования тут же начинается следующее) или в одиночном, когда каждый раз поступает команда на запуск. Кроме того, запуск режима ADC Noise Reduction инициирует начало преобразования, а с его окончанием МК автоматически "просыпается" и переходит к обработчику прерывания АЦП. Режим ADC Noise Reduction останавливает все схемы МК (кроме асинхронного таймера и WD-таймера) до окончания преобразования. Неудобство применения режима ADC Noise Reduction состоит в том, что при каждом измерении МК "умирает" не менее чем на 20 тактов (считая само преобразование, время "пробуждения" и выполнения команд инициализации режима), и если измерения производятся достаточно часто, это может мешать другим функциям

контроллера. Для моделей, где режима ADC Noise Reduction нет, можно использовать режим Idle, в котором, правда, останавливаются не все устройства (см. *главу 4*). В любом случае возиться с этими режимами целесообразно лишь тогда, когда, по крайней мере, и питание, и опорное напряжение АЦП подаются от отдельных прецизионных источников — иначе эти меры все равно не дадут нужного эффекта.

АЦП включается установкой бита ADEN (бит 7) в регистре управления АЦП, который в большинстве моделей Mega называется ADCSRA, а в некоторых других Mega (например, ATmega8), в "классическом" AT90S8515 и в моделях семейства Tiny называется просто ADCSR. Режим непрерывных измерений активизируется установкой бита ADFR (бит 5) этого же регистра. В ряде моделей Mega (к ним относится и употребляемый нами далее ATmega8535) этот бит носит наименование ADATE, и управление режимом работы производится сложнее: там добавляются несколько режимов запуска через различные прерывания (в т. ч. прерывание от компаратора, при наступлении различных событий от таймера и т. п.), и выбирать их следует, задавая биты ADTS регистра SFIOR, а установка бита ADATE разрешает запуск АЦП по этим событиям. Так как нулевые значения всех битов ADTS (по умолчанию) означают режим непрерывного преобразования, то в случае, когда вы их значения не трогали, функции битов ADATE и ADFR в других моделях будут совпадать.

Если выбран режим запуска не от внешнего источника, то преобразование запускается установкой бита ADCS (бит 6 того же регистра ADCSR/ADCSRA). При непрерывном режиме установка этого бита запустит первое преобразование, затем они будут автоматически повторяться. В режиме однократного преобразования, а также независимо от установленного режима при запуске через прерывания (в тех моделях, где это возможно) установка бита ADCS просто запускает одно преобразование. При наступлении прерывания, запускающего преобразование, бит ADCS устанавливается аппаратно. Отметим, что преобразование начинается по фронту первого тактового импульса (тактового сигнала АЦП, а не самого контроллера!) после установки ADCS. Само преобразование занимает 13 (или 14 для дифференциального входа) периодов тактового сигнала АЦП, кроме первого после включения АЦП преобразования, которое займет 25 тактов.

По окончании одиночного преобразования бит ADCS аппаратно сбрасывается. Кроме того, по окончании любого преобразования (и в одиночном, и в непрерывном режиме) устанавливается бит ADIF (бит 4, флаг прерывания). Разрешение прерывания АЦП осуществляется установкой бита ADIE (бит 3) все того же регистра ADCSR/ADCSRA.

Для работы с АЦП необходимо еще установить его тактовую частоту. Это делается тремя младшими битами регистра ADCSR/ADCSRA под названием

ADPS0..2. Коэффициент деления частоты тактового генератора МК устанавливается по степеням двойки, все нули в этих трех битах соответствуют коэффициенту 2, все единицы — 128. Напомним, что оптимальная частота преобразования лежит в диапазоне 50–200 кГц, так что, например, для тактовой частоты МК, равной 4 МГц, коэффициент может иметь значение только 32 (состояние битов ADPS0..2 = 101, частота 125 кГц) или 64 (состояние битов ADPS0..2 = 110, частота 62,5 кГц). При тактовой частоте 16 МГц в допустимый диапазон укладывается только коэффициент 128.

Выборка источника опорного напряжения производится битами REFS1..0 регистра ADMUX (старшие биты 7 и 6), причем их нулевое значение (по умолчанию) соответствует *внешнему* источнику. Напряжение этого внешнего источника может лежать в пределах от 2 В до напряжения питания аналоговой части AV_{cc} (а оно, в свою очередь, не должно отличаться от питания цифровой части более чем на 0,3 В в большую или меньшую сторону). Можно выбрать в качестве опорного и питание самой аналоговой части, причем двояким способом: либо просто соединить выводы AREF и AVCC микросхемы, либо установить биты REFS1..0 в состояние 01 (тогда соединение осуществляется внутренними схемами, но заметим, что внешний опорный источник при этом должен быть отключен). Предусмотрен и встроенный источник (задается REFS1..0 в состоянии 11, при этом к выводу AREF рекомендуется подключать фильтрующий конденсатор), имеющий номинальное напряжение 2,56 В с большим разбросом от 2,4 до 2,7 В.

Подробности

На практике в простых случаях, не требующих высокой точности преобразования, обычно имеется один источник, от которого питают и аналоговую, и цифровую части, а также формируют опорное напряжение, отфильтрованное через *RC*- или *LC*-фильтр (см. далее рис. 10.5). Использование внутреннего источника опорного напряжения из-за его нестабильности вряд ли заметно повысит точность преобразования, но может быть удобно для масштабирования измеряемой величины. В ряде аналоговых датчиков, которые возможно питать от напряжений 3–5 В, целесообразно питать всю аналоговую часть схемы отдельно — тогда погрешность можно снизить за счет того, что измерения станут относительными (если питание аналоговой части снизится или повысится, то пропорционально, скорее всего, изменится и измеряемое напряжение). Наконец, в случаях особо точных измерений следует предусмотреть внешний калиброванный прецизионный источник — например, MAX873, MAX875 или аналогичные.

В любом варианте организации питания желательно (а при отдельном аналоговом питании — обязательно), чтобы проводник "аналоговой земли", для которой в МК имеется отдельный вывод, соединялся с "цифровой землей" как можно ближе к источнику питания. Если невозможно протянуть его отдельным проводником прямо к источнику, то соединение "аналоговой" и "цифровой" "земель" следует производить непосредственно на контакте разъема питания платы, от которого эти "земли" должны расходиться отдельными проводниками.

Руководство Atmel также рекомендует вокруг выводов аналогового питания и выводов АЦП (они всегда идут подряд) на обратной по отношению к проводникам стороне платы делать площадку, "заливая" ее полностью "аналоговой землей".

Результат преобразования АЦП оказывается в регистрах ADCH:ADCL. Поскольку результат 10-разрядный, то по умолчанию старшие 6 битов в регистре ADCH оказываются равными нулю. Чтение этих регистров производится, начиная с младшего ADCL, после чего регистр ADCH блокируется, пока не будет прочитан. Следовательно, даже если момент между чтением регистров попал на фронт 14 (15) такта АЦП, когда данные в них должны меняться, значения прочитанной пары будут соответствовать друг другу, пусть и результат этого преобразования пропадет. В противоположном порядке читать эти регистры не рекомендуется. Но бит ADLAR (бит 5 регистра ADMUX) предоставляет интересную возможность: если его установить в 1, то результат преобразования в регистрах ADCH:ADCL выравнивается влево: бит 9 результата окажется в старшем бите ADCH, а незначащими будут младшие 6 битов регистра ADCL. В этом случае, если хватает 8-разрядного разрешения результата, можно прочесть только значение ADCH.

Выбор каналов и режимов их взаимодействия в АЦП производится пятью битами MUX0..4 в регистре ADMUX. В некоторых моделях (семейство Tiny) этих битов всего три (MUX0..2), а, например, в ATmega8 — четыре (MUX0..3), в зависимости от общего числа каналов. В любом случае их значения от 0 до максимального номера канала (которых в большинстве случаев 8, так что значения оказываются в пределах от 000 до 111, старшие биты, если они есть, равны 0) выбирают нужный канал в обычном (недифференциальном) режиме, когда измеряемое напряжение отсчитывается от "земли" (аналоговой). А последние два значения этих битов для семейства Mega (11110 и 11111 в большинстве моделей или 1110 и 1111 для ATmega8) выбирают режимы, когда вход АЦП подсоединяется к опорному источнику компаратора (1,22 В) или к "земле" соответственно, что может использоваться для автокалибровки устройства. В имеющих АЦП моделях Tiny (а также в "классическом" AT90S8535) такого режима нет.

Наконец, остальные комбинации разрядов MUX предназначены для установки различных дифференциальных режимов — в тех моделях, где они присутствуют, в других случаях эти биты зарезервированы (как в моделях Atmega8, ATmega163 и др.). В дифференциальном режиме АЦП измеряет напряжение между двумя выбранными выводами (например, между ADC0 и ADC1), причем не все выводы могут быть в таком режиме задействованы. В том числе дифференциальные входы АЦП можно подключать к одному и тому же входу для коррекции нуля. Дело в том, что в ряде моделей на входе АЦП имеет-

ся встроенный усилитель, с коэффициентом 1×, 10× и 200× (коэффициент выбирается теми же битами `MUX0..4`), и такой режим используется для его калибровки — в дальнейшем значение выхода при соединенных входах можно просто вычесть.

Входы и их режимы допускается переключать в любой момент, потому что эффект это даст все равно только после окончания текущего преобразования — так можно менять каналы "на ходу" в режиме непрерывного преобразования. Только при этом следует учесть, что в дифференциальном режиме из-за наличия усилителя показания установятся только через 125 мкс (т. е. примерно через 16 циклов непрерывного преобразования с частотой 125 кГц), результаты до истечения этого срока окажутся недостоверными. Почему-то для ATtiny15, где предусилитель имеет только два коэффициента (1× и 20×) это не оговаривается, зато указывается, что при переключении на дифференциальный режим первое измерение длится не 14, а 25 тактов, как при первом включении.

Для недифференциального режима АЦП, когда напряжение отсчитывается от "земли", результат преобразования определяется формулой: $K_a = 1024U_{вх}/U_{ref}$, где K_a — значение выходного кода АЦП, $U_{вх}$ и U_{ref} — входное и опорное напряжения. Дифференциальному измерению соответствует такая формула: $K_a = 512(U_{pos} - U_{neg})/V_{ref}$, где U_{pos} и U_{neg} — напряжения на положительном и отрицательном входах соответственно. Если напряжение на отрицательном входе больше, чем на положительном, то результат в дифференциальном режиме становится отрицательным и выражается в дополнительном коде от $200 (–512) до $3FF (–1). Реальная точность преобразования в дифференциальном режиме равна 8 разрядам.

Пример использования АЦП

Предположим, у нас имеется два датчика, например, температуры и давления, выдающих сигнал в пределах от 0 до 5 В. Необходимо измерять их значения не реже, чем один раз в несколько секунд и складывать в память. Потом эти данные будут извлекаться для индикации, которая осуществляется в отдельном временно́м цикле (так, как мы это делали в *главе 8*). Так как максимально возможная скорость измерений не требуется, целесообразно их проводить несколько раз (чем больше, тем лучше) и выводить усредненное значение.

Еще один канал АЦП мы займем измерением напряжения все той же резервной батарейки для мониторинга ее состояния. На этот раз мы можем более подробно индицировать ее состояние — не просто "исправна-неисправна", а выделить состояния "полностью исправна" (напряжение более ~3,4 В), "тре-

бует замены" (напряжение от ~2,7 до ~3,4 В) и "полностью неисправна" (напряжение менее ~2,7 В — сами уровни напряжения, естественно, могут быть и другими). Эти состояния будем индицировать двухцветным двухвыводным светодиодом (например, L117 — в прямоугольном, или L56 в круглом корпусе), который будет подсоединен к выводам PD7 и PC7. Если эти выводы в одном состоянии (например, оба в нулевом), то светодиод погашен, если единица на выводе PD7 — горит красный, если единица на выводе PC7 — горит зеленый (сравните с примером в *разделе "Команды проверки-пропуска" главы 6* — здесь алгоритм немного сложнее). Красный означает, что батарейка неисправна, зеленый — что полностью исправна, в состоянии "требует замены" светодиоды будут мигать попеременно.

Схема соединений, соответствующая этой задаче, с ориентировкой на ATmega8535, показана на рис. 10.5. Разряды PC0–PC6 служат для управления сегментами индикаторов (алгоритм изложен в *главе 8*, и здесь мы его разбирать не будем). Для коммутации разрядов индикаторов можно использовать свободные разряды порта B или D (на рис. 10.5 не показаны). Подключение батарейки аналогично рис. 10.2, поэтому сама она на рис. 10.5 также не показана.

В качестве датчика температуры (канал 0 АЦП) выберем полупроводниковый преобразователь с одним питанием LM50 фирмы National Semiconductor, который можно заменить без каких-либо доработок на TMP36 (Analog Devices) или на TC1047 (Microchip Technology). Выходной сигнал этого датчика при 0 °C равен 500 мВ, и имеет крутизну 10 мВ/° (с довольно большим разбросом, соответствующем ошибке примерно 2–3°, т. е. датчик необходимо калибровать). Таким образом, в диапазоне от –40 до +100 °C (рабочий диапазон LM50) напряжение будет меняться от 0,1 до 1,5 В, что соответствует изменению выходного кода в пределах от ~20 до ~300 с крутизной около 2 единиц кода на 1 °C.

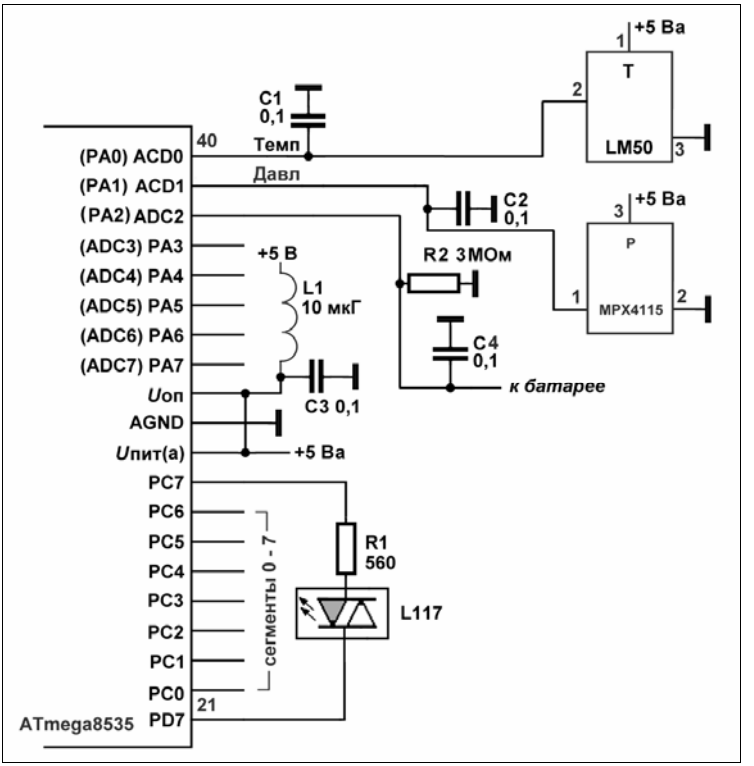

Рис. 10.5. Схема измерения температуры, давления и напряжения резервной батареи

Заметки на полях

Как видим, диапазон изменения немного превышает восемь разрядов, и есть довольно значительный запас, который бы позволил демонстрировать температуру с десятыми градуса (особенно если сократить диапазон). Именно на такой случай в АЦП предусмотрена возможность усиления входного сигнала. Мы могли бы установить усиление с коэффициентом 10, для чего, кроме всего прочего, пришлось бы переключить АЦП в дифференциальный режим. В принципе это удобно — если на отрицательный вход подать напряжение, соответствующее 0°, то можно получать температуру сразу со знаком. Но на практике в данной ситуации ничего не получается — после усиления в 10 раз нулевое значение (изначально 500 мВ) будет соответствовать максимуму диапазона напряжений, к тому же вся шкала (50 °С) оказывается слишком мала. Если задача позволяет ограничиться только положительными значениями температуры до 50°, то можно взять классический LM35 (у него 0 °С соответствует нулевому напряжению), и для него применить метод с включением усиления. Можно включить тот же LM35 по схеме со сдвигом питания, урезав шкалу в положительной области за счет отрицательной. Но все это полумеры — как видите, для под-

гонки шкалы подобных датчиков приходится добавлять внешние усилители, и мы не будем здесь на этом останавливаться.

Датчик атмосферного давления MPX4115 фирмы Motorola (канал 1 АЦП) имеет куда более удобную шкалу, в которой давлению от 15 до 115 кПа (от ~110 до ~860 мм рт. ст.) соответствует выходное напряжение от ~0,2 до ~4,8 В. Следовательно, мы используем выходной диапазон АЦП практически полностью, с крутизной примерно 1,3 единицы кода на 1 мм рт. ст., что нас устраивает с точки зрения разрешения.

Чтобы не отвлекаться на частности, расчет физических величин температуры и давления оставим за скобками (о нем поговорим немного в конце главы), а с напряжением батарейки (канал 2 АЦП) все просто: выберем в качестве опорного напряжение питания 5 В, т. е. столько придется на 1024 градации шкалы. Тогда например, значение 2,5 В будет соответствовать коду 512 (в предположении, что опорное напряжение равно точно 5 В). Так как ошибка самого опорного напряжения может быть достаточно велика (разброс выходного напряжения стабилизаторов типа LM2931 или LM78L05 на практике составляет от 4,9 до 5,1 В, а в принципе может быть и больше), а высокая точность нас тут не интересует, не нужно заниматься точной подгонкой результатов. Мы можем воспользоваться возможностью выравнивания результатов преобразования влево, и взять только старший байт, но это только усложнит логику программы — проще читать данные АЦП единообразно. Будем считать, что абсолютное значение напряжения с разрешением до двух знаков после запятой будет равно примерно половинному значению кода АЦП. Иначе говоря, выбранные нами границы напряжения выразятся так: 2,7 В — код 540 ($21C), 3,4 В — код 680 ($2A8).

Программа

Управлять измерениями и осуществлять динамическую индикацию (см. *главу 8*) будем через прерывания Timer0. В данном случае запустим эти прерывания с частотой немного менее 2 кГц (коэффициент деления частоты 1/8, частота "кварца" 4 МГц), что для динамической индикации удобно. Зарезервируем два регистра под счетчики, один из которых будет отсчитывать каждое 32-е прерывание, в котором мы будем АЦП запускать в однократном режиме к следующему разу и проверять, сколько раз мы уже измерили. Каждую величину будем измерять 64 раза, суммируя в обработчике прерывания АЦП эти значения в памяти для последующего усреднения, после каждого 64-го измерения в прерывании таймера будем окончательно обрабатывать данные. Такой режим даст нам 64 значения, равномерно распределенные по времени в течение чуть более одной секунды ($32 \times 64 = 2048$). После этого обрабаты-

ваем данные, переключаем АЦП на следующий канал и опять читаем 64 раза. Таким образом, каждую из трех величин мы обновляем примерно раз в три секунды, имея усредненное значение в течение одной секунды.

Каналы будем отсчитывать с помощью битов в регистре флагов (`Flag`) — единичное значение бита 0 будет соответствовать измерению температуры, бита 1 — измерению давления, бита 2 — измерению напряжения батареи. Еще один бит будет сигнализировать о состоянии батареи — единичное состояние бита 3 означает, что она "требует замены" (мигание из зеленого в красный). Бит 4 будет применяться для определения текущей ситуации в режиме мигания.

Листинг 10.6 содержит константы и переменные, которые потребуются в программе (в память мы будем здесь грузить "сырые" данные, о переводе в физические величины поговорим позднее).

Листинг 10.6

```
;кварц 4 МГц
.include "m8535def.inc"
;адреса SRAM старший байт адреса SRAM=0x01
.equ Tram = 0x0 ;0x0,0x1 — ст. и мл. байты T сумма
.equ Pram = 0x2 ;0x2,0x3 — ст. и мл. байты P сумма
.equ Bram = 0x4 ;0x4,0x5 — ст. и мл. байты батареи сумма
.equ Tres = 0x6 ;0x6,0x7 — ст. и мл. байты T рез. усреднения
.equ Pres = 0x8 ;0x8,0x9 — ст. и мл. байты T рез. усреднения
.equ Bres = 0x0A ;байт батареи рез. усреднения
. . . . .
.def AregH = r01 ;результат измерения, старший байт
.def AregL = r02  ;результат измерения, младший байт
.def temp = r16 ;рабочий регистр
.def temp1 = r17 ;рабочий регистр
.def count64 = r19;счетчик преобразований — до 64
.def countCyk = r20 ;счетчик циклов АЦП до 32
.def Flag = r21; регистр флагов
. . . . .
```

Инициализацию портов, регистров и памяти иллюстрирует листинг 10.7.

Листинг 10.7

```
ldi temp,0x80 ;PD7 — красный LED
out DDRD,temp
ldi temp,0xFF ;PC7- зеленый LED, ост. сегменты индикации
```

```
out DDRC,temp
clr countCyk
clr count64
sbr Flag,0x01 ;сначала пишем температуру
ldi r29,1 ;YH, пишем в RAM, начиная с адр. 01:00
;обнуление рабочих ячеек:
      clr temp
      ldi r28,Tram ;температура
          st Y+,temp
          st Y+,temp
          st Y+,temp ;давление
          st Y+,temp
      st Y+,temp ;батарея
      st Y,temp
. . . . .
```

Инициализация таймера показана в листинге 10.8.

Листинг 10.8

```
ldi temp,(1<<TOIE0) ;разр. прер. Timer0
out TIMSK,temp
ldi temp,255 ;очищаем прерывания
out TIFR,temp
ldi temp,0b00000010
out TCCR0,temp ;Timer0 вкл. 1:8 ~2000 Гц
. . . . .
```

Инициализация АЦП и сразу его первый запуск проиллюстрирована листингом 10.9.

Листинг 10.9

```
;start ADC 1/32 = 125 кГц; interrupt enable
ldi temp,1<<ADEN|1<<ADIE|1<<ADPS2|1<<ADPS0
out ADCSRA,temp
. . . . .
```

Не забудьте установить векторы прерываний компаратора и таймера. В прерывании АЦП мы производим только чтение данных и складываем сумму со старыми значениями в память по нужному адресу (листинг 10.10).

Листинг 10.10

```
ADC_INT: ;чтение АЦП по прерыванию
     ldi YH,1 ;старший SRAM
     sbrc Flag,0
         ldi r28,Tram ;установка адреса — T
     sbrc Flag,1
         ldi r28,Pram ;установка адреса — P
     sbrc Flag,2
         ldi r28,Bram ;установка адреса — Bat
     ld AregH,Y+ ;загружаем старое значение
     ld AregL,Y
     in temp1,ADCL;получаем младший ADC
       in temp,ADCH ;старший
          add     AregL,temp1 ;суммируем
          adc     AregH,temp
     dec r28 ;возвращаемся к адресу
     st Y+,AregH ;запоминаем сумму
          st Y,AregL ;в памяти
reti ;конец чтения ADC
```

Теперь самое главное: алгоритм переключения каналов и обработки данных в прерывании Timer0 (листинг 10.11).

Листинг 10.11

```
TIM0_INT: ;прерывание Timer0
. . . . .
<здесь динамическая индикация>
. . . . .
readADC: ;обработка АЦП
  inc countCyk
  sbrs countCyk,5 ;если бит 5 = 1, то прошло 32 прерывания
  reti ;иначе выход
  clr countCyk
  inc count64
  cpi count64,65 ;если прошло 64 чтения, то сразу на обработку
  breq endADC
  sbrc Flag,0  ;если бит 0 флага — будем читать температуру
  ldi temp,0
  sbrc Flag,1 ;если бит 1 флага — будем читать давление
  ldi temp,1
  sbrc Flag,2 ;если бит 2 флага — будем читать батарею
```

```
  ldi temp,2
  out ADMUX, temp ;установили АЦП канал
  sbi ADCSRA,ADSC  ;запуск нов. преобразования
  reti ;выход из прерывания таймера
endADC: ;обработка данных
;сначала, если необходимо, мигание ~1сек
       sbrs Flag,3 ;если флаг 3 стоит, надо мигать
       rjmp calcA ;иначе к расчету
       sbrs Flag,4 ;проверяем бит 4
     rjmp setR ;если сброшен — красным
       cbr Flag,0b00010000 ;иначе сбрасываем
     cbi PortD,7
       sbi  PortC,7 ;будем гореть зеленым
       rjmp calcA ;к расчету
setR:      sbr Flag,0b00010000 ;устанавливаем бит 4
     cbi PortC,7
     sbi PortD,7 ;горим красным
calcA:     ;расчет по 64 значениям
     clr count64 ;очистили счетчик
     sbrc Flag,0 ;узнаем текущий канал
         ldi r28,Tram ;установка адреса — T
     sbrc Flag,1
         ldi r28,Pram ;установка адреса — P
     sbrc Flag,2
         ldi r28,Bram ;установка адреса — Bat
     ld AregH,Y+ ;загрузка суммы из памяти
     ld AregL,Y
div64L:  ;деление на 64
      lsr AregH ;сдвинули старший
     ror AregL;сдвинули младший
     inc count64
     cpi count64,6
  brne div64L ;сдвинули-поделили на 64
  subi r28,1 ;возвращаемся к адресу старшего
  clr temp
  st Y+,temp
  st Y,temp ;очистили память для следующего цикла
  adiw r28,5; адрес результата на 5 больше
  st Y+,AregH ; запоминаем в памяти усредненное значение
  st Y,AregL
battery: sbrs Flag,2 ;расчет батареи
  rjmp tempr ;если бит 2 не установлен, то на температуру
  cbr Flag,0b00001000 ;гасим мигалку
```

```
  mov temp,AregL
  cpi temp,$1C
  ldi temp,$02
  cpc AregH,temp
  brsh ContC ;перейти, если больше 2,6 В
  cbi PortC,7
  sbi PortB,5 ; батарея в нуле, горим красным
  rjmp contPT
contC:     mov temp,AregL
  cpi temp,0xA8 ;3,3 В
  ldi temp,0x02
  cpc AregH,temp
  brsh contG ;перейти, если больше 3,3
  sbr Flag,0b00001000 ;мигать будем
  rjmp     contPT
contG:     cbi PortB,5
     sbi  PortC,7 ; норма, будем гореть зеленым
     rjmp     contPT
tempr:     sbrs Flag,0 ;расчет температуры
     rjmp prs ;переход к давлению
. . . . .
<здесь расчет физической величины T>
. . . . .
  rjmp     contPT
prs:  sbrs Flag,1 ;расчет давления
  rjmp contPT
. . . . .
<здесь расчет физической величины P>
. . . . .
contPT:
;установка флагов и переменных для следующего цикла
      clr count64
     sbrc Flag,0
     rjmp _F0
     sbrc Flag,1
     rjmp _F1
     cbr Flag,0x7
     sbr Flag,0x1 ;был бит 2, устанавливаем бит0 — температуру
       reti
_F0:
     cbr Flag,0x7
     sbr Flag,0x2 ;был бит 0, устанавливаем бит1 — давление
     reti
```

```
_F1:
     cbr Flag,0x7
     sbr Flag,0x4 ;был бит 1, устанавливаем бит2 — батарею
reti; конец прерывания Timer0
```

Как видите, процедура получилась довольно громоздкой, и можно осторожно задать вопрос: если мы сюда еще включим расчеты физических величин, которые занимают много времени, не выйдем ли за пределы интервала, отпущенного на прерывание (~500 мкс)? На самом деле этого не произойдет: ведь большинство прерываний пустые или почти пустые, лишь в одном из 2048 выполняется расчет значений, и каждый раз только для одной величины (а не сразу для всех). Если даже принять, что в каждом таком расчете участвует одна-две процедуры умножения (~100 тактов каждая) и еще пара таких процедур, как перевод в BCD, то весь расчет займет явно не более 500 тактов, т. е. менее 125 мкс — у нас имеется четырехкратный запас. И все же, если есть подозрение, что длительности прерывания не хватит, следует либо подсчитать время выполнения процедур более точно (что не всегда просто ввиду многовариантности), либо ориентироваться на худший вариант, и выбрать "кварц" с большей частотой.

Подробно о том, как рассчитывать физические величины, мы говорить не будем (см. предыдущий пример с компаратором — все аналогично, только коэффициенты, естественно, будут другими), есть только один интересный вопрос, касающийся знака температуры. В явном виде он здесь нигде не присутствует, что понятно, т. к. у всех датчиков линейная зависимость от абсолютной температуры (в градусах Кельвина) и ни про какие придуманные людьми отрицательные значения в Цельсиях они "не знают". Поэтому знак температуры придется определять явно, сравнением со значением кода при 0 °С (у нас эта "подставка" будет приблизительно равна 100, но в общем случае может быть и двухбайтовой).

Но этого мало, т. к. от нуля вниз абсолютное значение температуры (без учета знака) будет опять повышаться, а кода — снижаться. Следовательно, для получения верного результата выше нуля придется "подставку" при 0 °С вычитать из кода, а ниже 0 °С — из значения "подставки" вычитать код. Если "подставка" была загружена в регистры `KoeffH:KoeffL`, то весь алгоритм будет выглядеть так, как в листинге 10.12 (предполагаем, что к выводу PD6 подсоединен светодиод или сегмент, индицирующий знак минус для температуры; результаты АЦП, напомним, находятся в регистрах `AregH:AregL`).

Листинг 10.12

```
;учет знака:
       cp AregL,KoeffL ;сравниваем подставку с данными
       cpc AregH,KoeffH
```

```
       brsh b0 ;если данные больше, то переход
      sub KoeffL,AregL ;вычитаем из подставки данные
      sbc KoeffH,AregH
      mov AregL,KoeffL ;и загружаем их обратно
      mov AregH,KoeffH ;в регистры AregH:AregL
      sbi PortB,6 ;знак минус
      rjmp m0 ;к дальнейшему расчету
b0: ;если данные больше подставки, то знак +
      sub AregL,KoeffL ;вычитаем из данных подставку
      sbc AregH,KoeffH
      cbi PortB,6 ;гасим знак
m0: ;умножение на коэффициент крутизны характеристики
. . . . .
```

Обратите внимание, что здесь мы сначала оперировали со свободным членом — "подставкой", и лишь потом переходили к умножению на коэффициент крутизны характеристики. Конечно, можно выразить "подставку" и в натуральных величинах, и вычитать ее после умножения на крутизну, но в данном случае это гораздо менее удобно.

В итоге мы расправились с АЦП в довольно простом случае измерений по трем каналам при единообразных данных и без каких-либо требований к скорости проведения измерений (ясно, что при необходимости температуру и давление можно замерять и реже, чем раз в три секунды). Встречаются случаи намного более сложные — особенно, когда при всем этом еще необходимо учитывать энергосбережение. А сейчас мы перейдем к совсем другой теме — обмену данными между МК и "внешним миром", которому придется посвятить несколько глав.