Оуэн Бишоп

Настольная книга разработчика

# РОБОТОВ

Откройте для себя
мир робототехники
в ходе разработки
и программирования
собственных роботов

WWW.MK-PRESS.COM

# Robot Builder's Cookbook

Owen Bishop

ELSEVIER

**Elsevier Ltd.**
Boulevard, Langford Lane,
Kidlington, Oxford, OX5 1GB,
UK

Оуэн Бишоп

# Настольная книга разработчика роботов

Перевод с английского: В. В. Литвин

**Киев, "МК-Пресс"**
**СПб, "КОРОНА-ВЕК"**
**2010**

ББК 32.973-04
УДК 004.312
Б67

**Бишоп О.**

Б67 Настольная книга разработчика роботов. – К.: "МК-Пресс", СПб.: "КОРОНА-ВЕК", 2010. – 400с., ил.

ISBN 978-5-7931-0546-0 ("КОРОНА-ВЕК")
ISBN 978-966-8806-64-3 ("МК-Пресс")
ISBN 978-0-7506-6556-8 (англ.)

Эта книга представляет собой справочное руководство для тех, кто хочет научиться проектировать и конструировать роботов. Благодаря представленным в ней пошаговым инструкциям, вы быстро освоите методики создания забавных и захватывающих роботов. На основании своего обширного практического опыта автор открывает важные аспекты программирования, электроники и механики, характерные для робототехники. Поскольку все проекты основаны на использовании всемирно популярных микроконтроллеров PIC, методики программирования осваиваются быстро и безболезненно.

Данное руководство — идеальный вариант для новичков в сфере робототехники. Оно будет также полезно тем опытным разработчикам, которые хотят расширить свои познания в области программирования роботов. И безусловно, эта книга пригодится студентам, выполняющим практические задания по проектированию систем на основе микроконтроллеров.

**ББК 32.973-04**

*Главный редактор: Ю. А. Шпак*

Подписано в печать 25.09.2009. Формат 60 × 84 1/16. Бумага газетная. Печать офсетная.
Усл. печ. л. 23,3. Уч.-изд. л. 18,2. Тираж 2000 экз. Заказ №884

СПД Савченко Л.А., Украина, г. Киев, тел./факс: (044) 517-73-77; e-mail: info@mk-press.com.
Свидетельство о внесении субъекта издательского дела в Государственный реестр издателей, производителей и распространителей издательской продукции:
серия ДК №51582 от 28.11.2003г.



ISBN 978-5-7931-0546-0 ("КОРОНА-ВЕК")
ISBN 978-966-8806-64-3 ("МК-Пресс")
ISBN 978-0-7506-6556-8 (англ.)





# Содержание

# О книге...

Эта книга представляет собой практическое пособие по робототехнике, рассчитанное, в основном, на начинающих и малоопытных разработчиков. Она описывает механику конструкций роботов, принципы построения электронных схем и особенности программирования роботизированных систем.

Часть I книги содержит теорию, идеи и рекомендации для начинающих робототехников. Часть II посвящена краткому описанию микроконтроллеров PIC, на которых построены рассматриваемые далее проекты.

В части III представлен процесс проектирования, сборки и программирования пяти роботов различной степени сложности. Используемые спецификации — гибкие, с акцентом на ключевых характеристиках, благодаря чему проекты можно легко адаптировать к тем материалам и компонентам, которые окажутся в наличии у читателя. Каждое описание указывает путь к совершенствованию систем для получения широкого диапазона сложных и уникальных роботов.

Программы написаны на ассемблере MPASM для микроконтроллеров PIC, что позволяет изменять, улучшать и расширять их. Представленные в книге листинги сопровождаются подробными комментариями и блок-схемами алгоритмов. Последние позволяют без проблем создавать аналогичные программы на одном из диалектов языков Basic и C.

# Введение

Первый вопрос, на который необходимо дать ответ, звучит: "Какой тип робота требуется создать?". Обычно, когда люди слышат слово "робот", многие из них сразу представляют себе R2-D2 из "Звездных войн" или кадры из фильма "Я робот". В определенном смысле эти роботы напоминают людей, но не во всем. Существует много разновидностей роботов. Отдельную большую группу составляют **мобильные роботы**, которых иногда еще называют **мобильными платформами**. К этой категории, в частности, относятся и вышеупомянутые человекоподобные роботы, а также широкий диапазон конструкций, имитирующих животных (рис. 0.1). Некоторые из них перемещаются на шести ногах, подобно насекомым, другие же прыгают, как лягушки. Существуют также более полезные мобильные роботы, которые снуют по дому, подметая пол, или же по цехам фабрики, доставляя детали на рабочие места. Такие устройства редко напоминают людей — они просто перемещаются и выполняют определенную задачу.

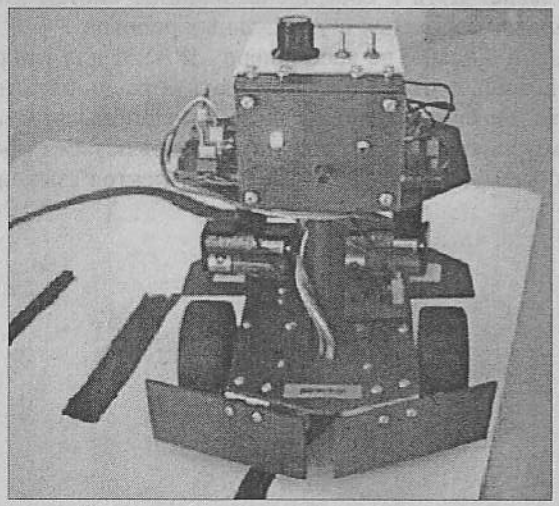


**Рис. 0.1.** Мобильные роботы напоминают людей лишь отдаленно, а многие — вообще не похожи на человека



Тем, кто только начинает свой путь в робототехнике, рекомендую проект недорогого мобильного робота (см. главу 6, "Скутер").

Люди способны выполнять большой круг задач, однако большинство роботов не настолько универсальны. В промышленности их используют для очень ограниченного числа операций, выполняемых по многу часов в день без свойственных человеку усталости, раздражения и ошибок. К этой категории относятся **манипуляторы**, основание которых обычно монтируют неподвижно. Они особенно удобны для выполнения утомительных или монотонных операций в промышленности (рис. 0.2). Манипуляторы также используют в средах, вредных или опасных для людей.

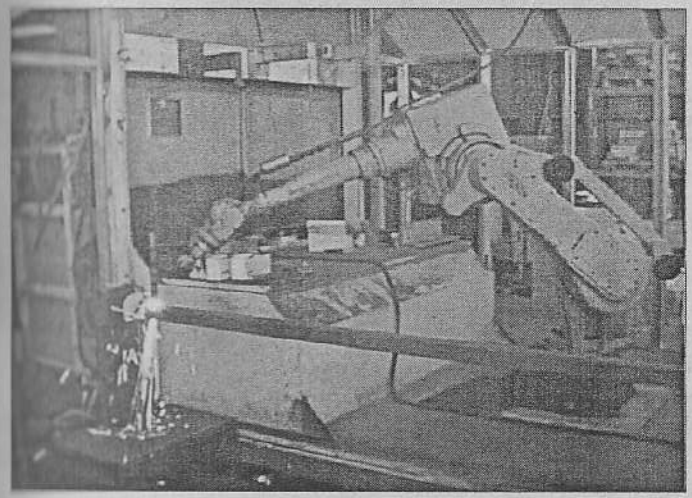


Рис. 0.2. На конце консоли этого робота установлен сварочный аппарат. Манипулятор прикреплен к полу, однако может переместить сварочную горелку практически в любую точку мастерской. При необходимости его можно оснастить и другим инструментом

Для подъема и точной транспортировки тяжестей используют **портальные роботы** (рис. 0.3). Их основной недостаток заключается в том, что они обычно — немобильные, в силу чего расстояние переноса грузов ограничено. В этой книге рассматривается портальный робот для поднятия и перестановки легких предметов (например, шахматных фигур).

Так каким же должен быть робот: мобильным или портальным? Какой бы тип вы ни выбрали, процесс проектирования состоит из одних и тех же этапов, рассмотренных далее в этой главе.

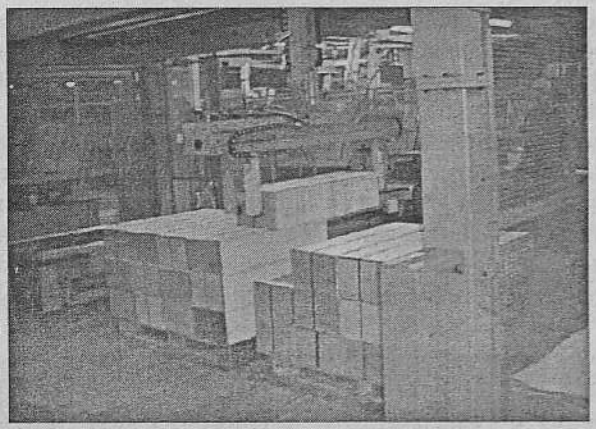


**Рис. 0.3.** Этот портальный робот в типографии собирает блоки отпечатанных листов, готовых для упаковки. Складывая блоки, он подсчитывает их, формируя пакеты определенного размера. Обратите внимание, какая у этого робота мощная крепежная колонна (изображена на переднем плане)

## Этапы создания робота

Определившись с типом создаваемого робота, на следующем этапе необходимо определить его первую спецификацию. Впрочем, в будущем ее, скорее всего, придется пересмотреть (по крайней мере — мелкие детали).

Для начала составьте перечень операций, которые должен выполнять робот, опишите его структуру, а также — какие электронные схемы он будет использовать. Соответствующие идеи можно почерпнуть в частях I и II данного руководства (рис. 0.4 – 0.7).

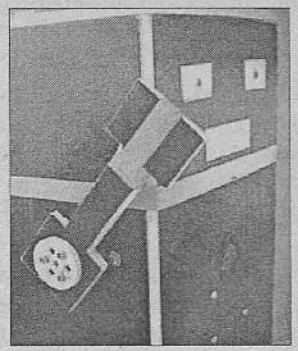


**Рис. 0.4.** В главе 1, "Поведение роботов", описано, что роботы могут делать



**Рис. 0.5.** В главе 2, "Механика роботов", рассмотрены структурные схемы, материалы, готовые компоненты, инструменты

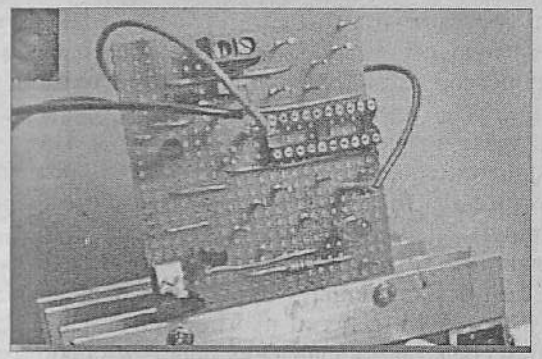


**Рис. 0.6.** Глава 3, "Электроника роботов", посвящена компонентам, датчикам, исполнительным механизмам и схемам управления

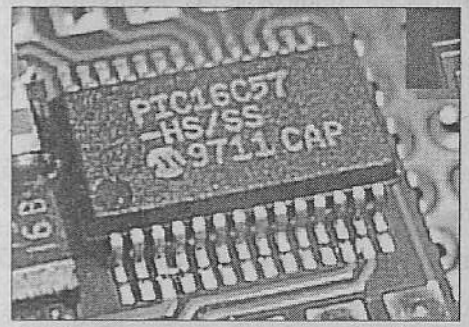



**Рис. 0.7.** В главах 4 и 5 рассмотрены микроконтроллеры PIC, их взаимодействие с электроникой и программирование для управления действиями робота

Данная книга — справочное руководство. Совсем не обязательно читать ее целиком от начала главы 2 до окончания главы 5. Просто просмотрите их, чтобы определиться с вопросами, представляющими интерес. Внимательно изучите соответствующие подразделы и сформируйте конечную спецификацию.

## Управление роботами

Первые роботы (тогда они назывались "автоматами") были чисто механическими и управлялись часовыми механизмами или паровыми двигателями. Появление электроники значительно расширило диапазон применения роботов. Начали формулироваться современные концепции робототехники. Значительным "прорывом" стало размещение сложных цифровых схем на одном кристалле. Так появились **микропроцессоры**, способные выполнять миллионы операций в секунду. Микропроцессоры широко используются в компьютерах, роботах, а также многих других устройствах, требующих высокоскоростной цифровой обработки данных. Микропроцессор не может работать изолированно. Должны присутствовать и другие электронные устройства: микросхемы памяти, порты ввода-вывода и генератор тактовой частоты. Платы, содержащие элементы микропроцессорной системы, — обычно большие и сложные (слишком сложные для того, чтобы их мог разработать средний энтузиаст робототехники).

Затем появились **микроконтроллеры** — "компьютеры на одном кристалле". Быстродействующие, простые в подключении к другим электронным компонентам, легко программируемые и недорогие, они стали идеальным средством управления простыми роботами (рис. 0.8).

Хотя выпуском микроконтроллеров занимаются сразу несколько компаний, устройства PIC производства корпорации Microchip Technology, пожалуй, — наиболее популярные среди радиолюбителей (да и во многих профессиональных областях). С точки зрения проектирования роботов очень важно правильно выбрать тип микроконтроллера PIC. Этот вопрос рассматривается в главе 4, "Использование микроконтроллеров PIC". Так, для проектов, описанных в части III книги используется модель PIC16F690, однако с тем же успехом можно было бы воспользоваться и некоторыми другими представителями PIC.

## Программирование микроконтроллеров PIC

Микроконтроллер работает по **программе**, которая хранится в цифровой форме в оперативной памяти в виде кода, называемого **машинным**. Этот код очень трудно писать вручную, но, к счастью, на помощь



приходит компьютер. С помощью специального программного обеспечения программу вводят в виде последовательности понятных команд (или **мнемоник**), которые контроллер должен выполнить. Затем программное обеспечение **транслирует** эти мнемоники в машинные коды.

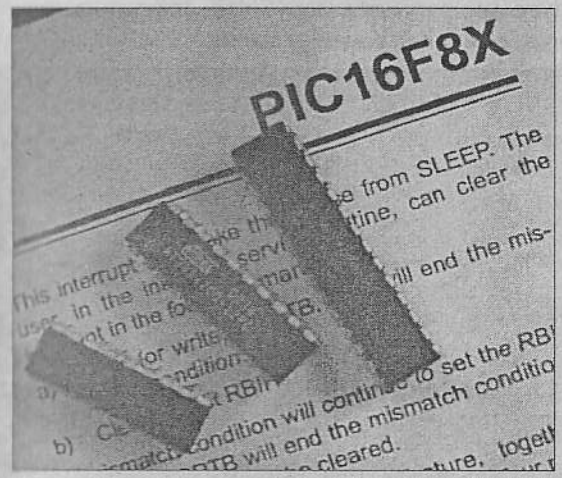



Рис. 0.8. Подборка микроконтроллеров (слева направо): PIC16F84 (18 выводов) от компании Microchip, AT90S1200 (20 выводов) от компании Atmel и PIC16F87 (28 выводов). Техническое описание на заднем плане было загружено бесплатно с сайта Microchip

С помощью **программатора** (рис. 0.9), транслированные машинные коды копируются из ПК в память микроконтроллера. Программатор обычно оснащен несколькими светодиодами и кнопками для контроля входных и выходных каналов PIC в ходе выполнения программы.

Микроконтроллеры PIC, рекомендуемые к использованию в проектах, рассмотренных в данной книге, содержат **флэш-память**. Этот вид памяти используется также в цифровых фотоаппаратах для хранения изображений. Преимущество флэш-памяти заключается в том, что она может перепрограммироваться по крайней мере 100 раз, поэтому является идеальным вариантом при разработке программного обеспечения для роботов. Программу следует вводить по частям, с последующим тестированием каждого фрагмента. Позднее, такие фрагменты могут быть добавлены или стерты, если в них окажется что-то не так. Разрабатываемая программа может быть даже полностью заменена совершенно новой.

Как уже упоминалось выше, нет необходимости создавать программу в машинных кодах, в которых она в конечном счете сохраняется. Вместо этого, программист использует язык ассемблера. Все микро-

контроллеры PIC поддерживают один и том же ассемблер, реализующий только 35 различных команд. Таким образом, изучить его не составляет труда. Ассемблер — это язык, в котором каждой команде соответствует только одно действие. Таким образом, программист шаг ша шагом сообщает микроконтроллеру, что следует делать. Такие программы — простые в анализе и понимании.

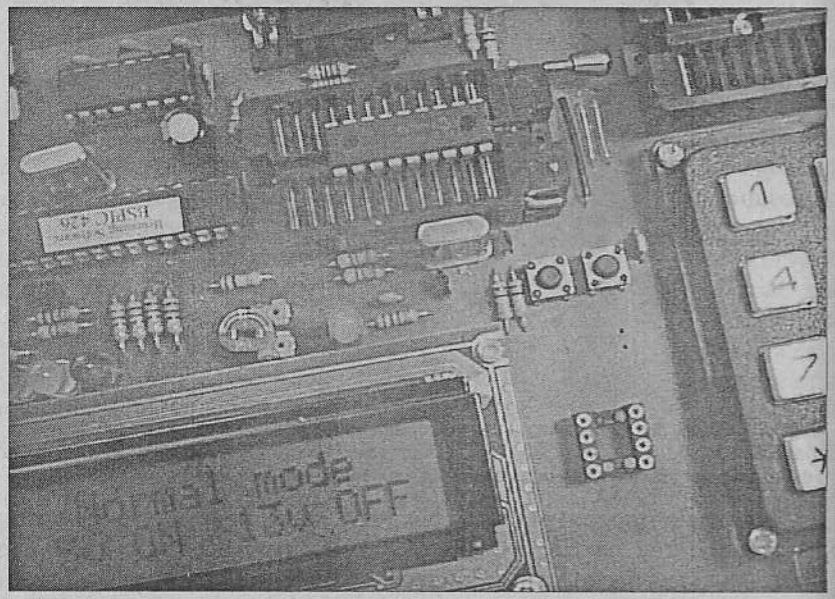


**Рис. 0.9.** PIC (вверху в центре) программируется в программаторе. Программатор связан с ПК, в котором выполняется специальное программное обеспечение. Индикатор программатора информирует пользователя о ходе операции.

Некоторые считают ассемблер сложным языком, поскольку микроконтроллер инструктируется очень мелкими шажками. Многие программисты не привыкли думать подобным образом и предпочитают использовать шаги побольше. Для этого применяется один из языков высокого уровня, наподобие Basic и C. Команды, написанные на этих языках, напоминают обычную английскую речь. Это упрощает программирование, однако требует пристального внимания к синтаксису команд, иначе компьютер может не понять программу.

Объем машинного кода, получаемого из команд высокого уровня с помощью специальной программы, называемой **компилятором**, обычно значительно больше, чем в случае эквивалентной программы на ассемблере. Для хранения такой программы требуется больше памяти и,



кроме того, она будет работать медленнее ассемблерной. Впрочем, для роботов, описанных в данной книге, это — не проблема, поскольку используемые в них программы — короткие, а высокое быстродействие не требуется.

Те, кто вообще не хочет ничего программировать, могут воспользоваться готовыми файлами с машинными кодами (`.hex`) на прилагаемом к книге компакт-диске.

## Имитация PIC

В состав программного обеспечения, поставляемого с программатором, обычно входит собственный ассемблер, а также, возможно, — встроенный компилятор Basic или C. Программы вводят на компьютере в любом текстовом редакторе (например, блокноте Windows). Затем наступает черед **имитатора** — программной оболочки, работающей на ПК, которая имитирует поведение микроконтроллера PIC. При этом программа может выполняться в реальном режиме времени или же пошагово, строка за строкой.

Во время работы имитатора специальные панели на экране компьютера отображают содержимое всех регистров имитируемого микроконтроллера PIC. Благодаря этому, легко увидеть, что происходит, и работает ли программа, как это было задумано. На экране также отображается содержимое памяти PIC. Кроме того, в более сложных имитаторах можно настраивать виртуальные входные и выходные устройства (светодиоды, цифровые индикаторы, кнопки) и наблюдать, как имитируемый микроконтроллер с ними взаимодействует.

Далее наступает этап поиска и коррекции ошибок. Когда все, наконец, проверено, транслированная программа загружается в PIC. Ассемблерные команды преобразуются в машинные коды и записываются в память программ микроконтроллера. Оказавшись внутри PIC, программа может оставаться там годами (или пока не будет изменена). После этого остается только подключить микроконтроллер в разъем в схеме робота и проверить его работоспособность.

# Часть I

# Основы робототехники

***В этой части...***

- Поведение роботов
- Механика роботов
- Электроника роботов

# Глава 1. Поведение роботов

Поведение робота определяется его типом. Мы начнем с мобильных роботов, а о портальных поговорим чуть позже. В завершение главы будут рассмотрены виды активности, общие для всех видов роботов.

## Поведение мобильных роботов

### Передвижение

По определению, все мобильные роботы должны быть способны передвигаться в прямом и обратном направлениях, а также — поворачиваться влево и вправо. Роботы зачастую работают в ограниченном пространстве, поэтому для них также было бы желательно уметь поворачиваться на месте. Переменная скорость менее важна, и обычно в ней нет необходимости.

Например, робот "Искатель", рассмотренный в главе 9 (рис. 1.1) движется на трех колесах. Два из них (левое и правое) — приводные, каждое — с собственным электродвигателем. Третье колесо — это ролик, используемый для устойчивости.

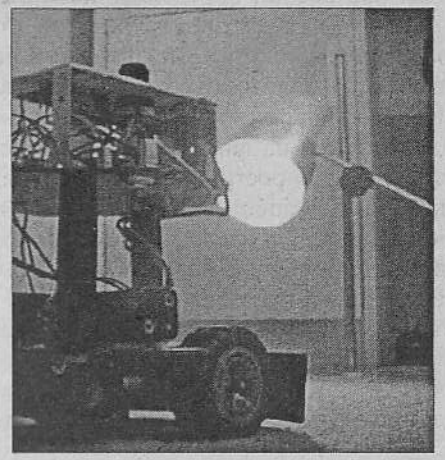


**Рис. 1.1.** Робот "Искатель" движется на источник света

Робот "Скутер" (см. главу 6) также оснащен тремя колесами, однако использует только один электродвигатель. Его движения несколько хаотичны, зато он недорог и прост в сборке.

### Обнаружение света и реакция на него

Зрение, наверное, — самое важное из всех человеческих чувств. Это же можно сказать и о мобильных роботах. Некоторые из них способны определить источник света на расстоянии в несколько метров и двигаться к нему (см. рис. 1.1). Или, наоборот, предпочитает двигаться в противоположном направлении, чтобы скрыться в безопасном темном углу.

Важная особенность света заключается в том, что его можно обнаруживать на расстоянии. Это делает его идеальным фактором для работы дистанционных датчиков. Впрочем, здесь возникает одна проблема: датчики могут быть введены в заблуждение комнатным или солнечным освещением. Одним из способов решить эту проблему являются импульсные источники света.

### Обход преград

Другое использование света связано с определением преград. Датчики приближения сообщают роботу о близости (но не касании) некоторого объекта. Под "объектом" подразумеваются неподвижные преграды, наподобие стен или мебели. При использовании датчиков приближения источник света вмонтирован в робота и обычно ориентирован вперед.

Светочувствительный датчик определяет наличие близко расположенного объекта, реагируя на свет, отраженный от этого объекта. Если интенсивность отраженного света превышает некоторое пороговое значение, то робот знает, что поблизости что-то находится. Светочувствительный датчик, направленный в сторону, может использоваться для того, чтобы робот удерживался на фиксированном расстоянии от стены. Движение вдоль стены — распространенный тип поведения робота, который часто используется для определения пути в лабиринте.

Альтернативой свету для определения факта приближения к объекту является ультразвук. Он требует более сложных электронных схем, однако на него не влияет внешнее освещение.

Ультразвуковой датчик можно запрограммировать на измерение расстояния, благодаря чему робот может распознавать окружающую обстановку и легко обходит преграды. Впрочем, хотя такое использование ультразвука и представляет экспериментальный интерес, его нельзя назвать безотказным.



### Распознавание касаний

Под касанием подразумевается физический контакт робота с препятствием, наподобие какого-либо массивного объекта или стены. Обычно, роботы оснащены бамперами или проволочными "антеннами", ориентированными таким образом, что касаться близлежащих предметов. Типичная реакция робота на касание — небольшой откат назад, поворот влево или вправо на незначительный угол и повторная попытка проехать вперед. При наличии пары передних бамперов (правого и левого) робот может выбрать наилучшее направление движения. Боковые же бамперы позволяют реализовать следование робота вдоль стены вместо датчиков приближения (рис. 1.2).

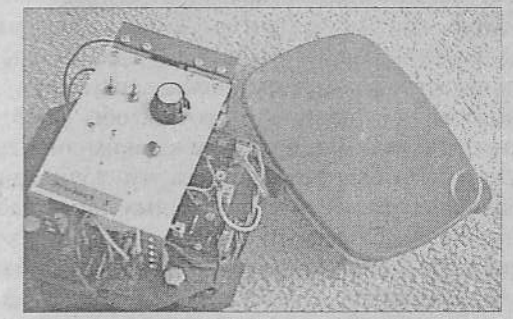


Рис. 1.2. Робот определил наличие коробки в результате касания ее правым бампером

К другим сферам применения датчиков касания относится расчистка некоторой области от легких объектов, а также поиск и подъем таких объектов. Факт контакта с небольшим объектом обычно можно определить по характеру движения робота. Если приводные двигатели включены, однако приводные колеса не поворачиваются, то робот, скорее всего, уперся в неподвижную преграду. Для того чтобы определить, поворачиваются колеса или нет, используется тахометр.

Еще один особый вид контактного поведения робота — следовании линии, нанесенной на поверхность движения. Для этого необходимы два простых светочувствительных датчика. Программирование в данном случае также не составляет труда. Кроме того, такая методика обеспечения пути робота из одной точки в другую — наиболее надежная.

### Взаимодействие

В большинстве случаев роботы должны взаимодействовать с людьми, те же из них, которые запрограммированы для игр, только этим

и занимаются. Например, робот может посылать сообщения человеку, мигая светодиодами или подавая звуковые сигналы.

Связь в обратном направлении обычно сводится к нажатию кнопки или переключению переключателя. Еще одна методика подразумевает использование датчиков, срабатывающих на звук.

Радиосвязь — способ взаимодействия с другими роботами для обмена информацией и координирования действий.

### Навигация

Если робот — мобильный, то ему желательно знать, где он находится. На практике это реализовать не так просто, как кажется на первый взгляд. К базовыми методам навигации относятся следование линии, движение вдоль стены и ориентирование на источник света. Они требуют минимального числа датчиков, наиболее просты в программировании, и в большинстве случаев их вполне достаточно.

Однако некоторые операции требуют, чтобы робот перемещался в "свободном пространстве", без привязки к каким-либо линиям, стенам или маякам. Можно было бы предположить, что точное позиционирование робота обеспечивает включение приводных двигателей на точно заданные отрезки времени, однако на практике такой подход не работает по одной единственной причине: два двигателя не вращаются в *точности* с одинаковой скоростью, даже если они — одного и того же типа. Поэтому, когда оба двигателя вращаются вперед, робот при движении слегка отклоняется влево или вправо, и в точности контролировать угол такого поворота не представляется возможным. Подобные погрешности накапливаются, и вскоре робот полностью теряет ориентацию.

Различия в работе двигателей можно нейтрализовать несколькими способами. Один из них заключается в использовании тахометра для подсчета полных и частичных оборотов каждого приводного колеса. Другой способ — задействовать вместо обычных двигателей постоянного тока шаговые. Однако, даже эти меры не всегда полностью решают проблему. В зависимости от характера поверхности и шин неизбежно присутствует небольшое проскальзывание. Оно обычно возникает в начале движения, при остановке и повороте, причем данная погрешность накапливается. И с этим ничего нельзя поделать.

Оптимальное решение в данном случае — периодическое определение роботом его местоположения, что позволит избежать накопления погрешности. Недостаток такого подхода заключается в необходимости использовать три световых маяка, что усложняет настройку системы. Кроме того, здесь должна быть задействована сложная математика.



Одним из современных решений данной проблемы является магнитный компас. Недорогие компасы с электронным выводом предлагаются некоторыми поставщиками робототехнических изделий. К сожалению, они недостаточно точны, однако представляют собой интересный материал для экспериментов.

Наиболее практичным решением будет портальный робот, которому посвящен следующий раздел.

## Поведение портальных роботов

Портальные роботы работают в четко определенной прямоугольной области. Они берут объекты в любой точке данной области и опускают их в другой точке этой же области. Рабочий инструмент такого робота (обычно — схват) подвешен на небольшой раме-вагонетке, и может опускаться и подниматься. Рама оснащена колесами и движется на паре рельсов, обеспечивающих перемещение от одного края рабочей области к другому. Эта пара рельсов закреплена на большей раме, расположенной под прямым углом к первой паре, в результате чего меньшая рама может перемещаться в любую точку рабочей области. Таким образом, позиция рабочего инструмента определяется двумя координатами: $x$ и $y$.

Разработать датчики, считывающие координаты $x$ и $y$, очень легко, и потому портальные роботы гораздо проще в программировании задач, требующих точной навигации.

Портальные роботы используются в промышленности для переноса очень тяжелых грузов. Их любительские версии решают задачи, требующие меньших физических усилий. Например, они прекрасно подходят для настольных игр типа шахмат, шашек и т.д.

Подобно мобильным, портальные роботы можно запрограммировать на прохождение лабиринта (рис. 1.3), однако мобильные роботы могут терять ориентацию, а портальные — никогда ввиду точного позиционирования подвижных рам за счет отслеживания координат $x$ и $y$.

При прохождении лабиринта портальный робот не движется по коридорам, как мобильный, а анализирует сверху нарисованное на бумаге или картоне изображение. Текущая позиция определяется по тонкому лазерному лучу, проецируемому из рамы.

## Обратная связь

В качестве примера обратной связи, можно привести обычный бытовой холодильник. Когда температура внутри него повышается выше заданного уровня, автоматически включается насос холодильного агрегата. Он остается включенным до тех пор, пока температура не упадет

до определенной отметки. Таким образом температура внутри холодильника удерживается в узком диапазоне значений.

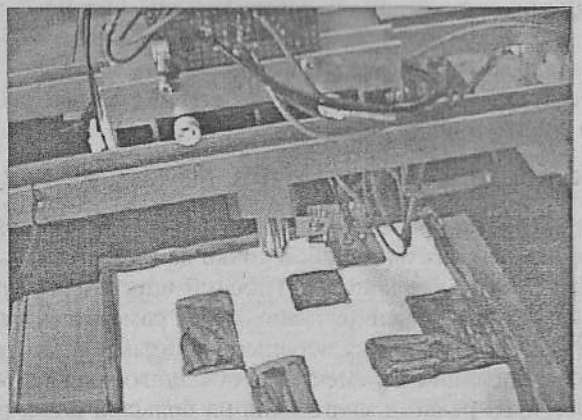


**Рис. 1.3.** Портальный робот проходит лабиринт, используя для отслеживания пути лазерный указатель

Такую обратную связь называют **отрицательной**, поскольку *увеличение* температуры приводит к *охлаждению* камеры. Примером из робототехники может служить схема регулятора скорости вращения двигателя на операционном усилителе (см главу 3, "Электроника роботов"). Схема регулятора скорости использует электронные аппаратные средства, которые обеспечивают сигнал обратной связи и формируют реакцию на него.

Обратную связь можно также регулировать программно. Представьте себе мобильного робота, который двигается вдоль стены, расположенной слева от него. Он оснащен инфракрасным светодиодом, направленным на стену, и инфракрасным датчиком, принимающим отраженные инфракрасные лучи. Программа поведения робота составлена таким образом, чтобы количество отраженных инфракрасных лучей поддерживалось на постоянном уровне. Это будет удерживать робота на одном и том же расстоянии от стены.

Когда робот приближается к стене, количество отраженного инфракрасного света увеличивается. Датчик распознает это, и программа реагирует поворотом робота вправо, что заставит его отодвинуться от стены. Прямо противоположное происходит, когда робот отклоняется вправо.

Типичная функция отрицательной обратной связи — обеспечение *постоянства* показателей. Она дает *стабильность*. Существует также



**положительная** обратная связь. Предположим, что в случае с описанным выше роботом, двигающимся вдоль стены, выходные линии, ведущие к двигателям, были по ошибке перепутаны. В этом случае малейшее отклонение от заданного значения расстояния приведет к драматическим результатам. Робот будет или все дальше удаляться от стены, или врежется в нее. Положительная обратная связь дает *нестабильность* — именно то, чего в поведении роботов следует избегать.

Существует и еще один тип обратной связи, очень важный в робототехнике. Например, рассмотрим случай, когда бульдозерообразный мобильный робот имеет перед собой эжектор для игры в "футбол". Обычно он высоко поднят, однако, когда робот видит перед собой мяч, опускается почти до земли. Эжектор должно быть близок к поверхности, но не должен касаться ее, чтобы не цепляться за неровности. Для этого он поднимается или опускается двигателем, который наматывает или разматывает намотанный на ось шнур (рис. 1.4).

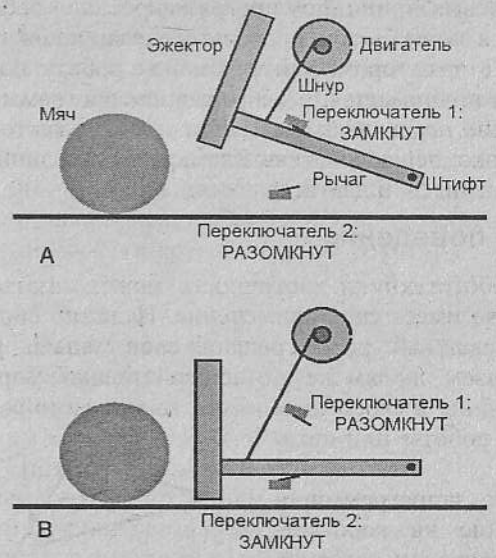



**Рис. 1.4.** Механизм поднятия и опускания эжектора, прикрепленного к передней части робота (сам робот и поддерживающие конструкции не показаны)

В данном случае используются микропереключатели, наиболее подходящие для подобных механизмов. Если двигатель наматывает шнур (вращение по часовой стрелке на рис. 1.4), то эжектор поднимается вплоть до касания опорным рычагом переключателя 1. Это замыкает переключатель и передает в микроконтроллер сигнал отключения дви-

гателя. Без подобной системы двигатель продолжал бы вращаться вплоть до повреждения эжектора или обрыва шнура.

Механизм оснащен и вторым переключателем, обнаруживающим момент опускания эжектора до положения над самой землей. Подобные переключатели, используемые для обнаружения чрезмерных перемещений механизмов, называют **концевыми**.

Аналогичным образом, отслеживание траектории подразумевает движение робота вперед при постоянном опросе контроле датчиков через малые интервалы времени. Даже на прямой линии могут встречаться неровности, заставляющие робота отклониться от намеченного курса. В таком случае срабатывает отрицательная обратная связь, заставляя робота вернуться обратно на траекторию.

### Опрос выходного сигнала

Пример с обратной связью от концевых выключателей иллюстрирует один из ключевых принципов программирования роботов: "Скажите ему, что делать, а затем быстро проверьте, сделал ли он это".

В примере с эжектором, дайте команду роботу поднять эжектор, а затем, пока он поднимается, в непрерывном программном цикле проверяйте состояние переключателя 1. Как только эжектор будет поднят достаточно высоко, переключатель 1 замкнется, входной сигнал микроконтроллера изменится, и двигатель остановится.

## Хаотическое поведение

В сфере робототехники хаотичность может казаться неуместной, однако она также имеет свое применение. Надежно спроектированный и запрограммированный робот решает свои задачи упорядоченным и негибким образом, людям же это не свойственно. Впрочем, если они слишком постоянны в своем поведении, то мы часто говорим, что они действуют, как роботы или автоматы.

Если робот запрограммирован на движение на короткие, случайно выбираемые расстояния с последующим поворотом на случайный угол, и будет выполнять эту программу неопределенно долго, то полученная траектория окажется совершенно хаотичной (рис. 1.5). В этом случае говорят, что робот буквально двигается по методу Монте-Карло. Слово "буквально" в данном случае используется потому, что термин "движение по методу Монте-Карло" применим и к другим случайным последовательностям, которые фактически никак не связаны с движением.

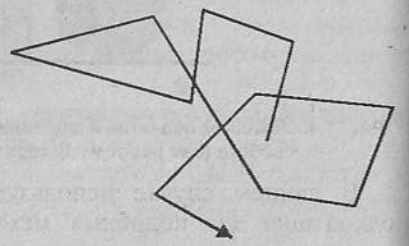


**Рис. 1.5.** Движение по методу Монте-Карло обычно приводит к нахождению примерно в одной и той же области



Обычно мы не преследуем цель достижения полной хаотичности. Например, если робот на своем пути встречает препятствие, он отъезжает назад, слегка поворачивается, чтобы объехать преграду, а затем продолжает движение вперед. Остановка, откат и поворот — это фиксированные реакции, однако выбор направления поворота выбирается случайным образом. Мы не можем предсказать, куда робот повернет в следующий раз, и это вносит в поведение робота небольшую хаотичность.

Хаотическое поведение реализуют программно с помощью процедуры генерирования случайных чисел. По правде говоря, выдаваемое ею значение — не идеально случайное, а представляет собой предсказуемый ряд значений, который повторяется через столь длительный интервал времени, что кажется случайным. Фактически, такие последовательности — **псевдослучайные**.

Хаотичность имеет и другое, более серьезное применение. Робота, который ищет выход в лабиринте, можно запрограммировать на случайный выбор направления на каждом перекрестке. Если он также запрограммирован на запоминание каждого сделанного выбора и определение, какой из них привел к цели, то он может в конечном счете научиться правильно проходить лабиринт. Это — основа **метода проб и ошибок**, применимого и к другим задачам обучения.

## Поглощение

Представим себе мобильного робота, запрограммированного на движение в направлении источника света. По достижении точки A (рис. 1.6) его датчик приближения обнаруживает преграду на пути к источнику света. Робот немедленно останавливается и переходит к этапу обхода препятствия. В таком случае движение к цели **поглощается** обходом. Робот поворачивается и следует к точке B. Однако он опять наталкивается на препятствие, поэтому поворачивается еще раз и двигается

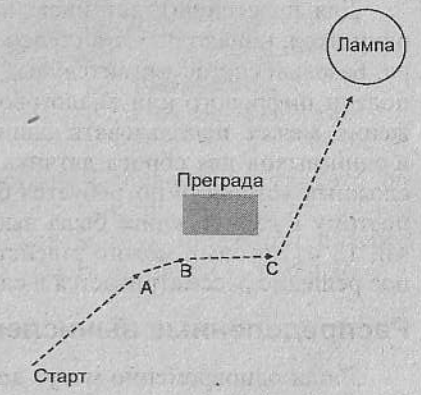



**Рис. 1.6.** Пример поглощения

к точке С. Преграда обойдена. В точке С открывается беспрепятственный путь к источнику света, и робот возобновляет движение к цели.

Поглощение одной фазы поведения другой используется как методика программирования всех типов роботов. В конце концов, такое поведение очень напоминает человеческое. Например, если во время обеда зазвонит телефон, то мы прекращаем есть, отвечаем на телефонный звонок, а потом опять возвращаемся к еде.

## Требования ко входам и выходам

Мы уже почти подошли к концу главы 1, и читатели, наверное, уже разработали внушительную спецификацию своего робота. Обычно в ней перечислено немало датчиков и исполнительных механизмов, однако каждый из таких компонентов требует одного или нескольких выводов микроконтроллера. Наступил момент критической оценки нашего проекта на предмет его выполнимости. Возможно, микроконтроллер PIC, выбранный для проекта, не предоставляет достаточного числа входов и выходов.

В качестве примера возьмем PIC16F84 — один из наиболее популярных микроконтроллеров семейства PIC. Он оснащен 13 каналами ввода-вывода. Достаточно ли этого? Тринадцать выводов — вроде бы, немало, однако робот танкового типа (с двумя приводами) требует четырех выходов для управления двигателями. Еще по одному выходу выделяется под каждый светодиод. Включение сирены — еще один выходной канал. Возможно также, что робот оснащен схватом, для которого требуется по крайне мере один канал. Итого, мы уже использовали семь выходов. Для датчиков осталось всего шесть каналов!

Для простейших датчиков, наподобие бамперов, требуется только один вход, однако робот в общем случае содержит более одного бампера. Базовый светочувствительный датчик нуждается в одном канале для подачи цифрового или аналогового входного сигнала. Датчик приближения может использовать один канал для уведомления о преграде и один выход для сброса датчика. Итак, все выводы заняты! Для более сложных устройств потребуется больше каналов ввода-вывода. Именно поэтому в данной книге была выбрана модель PIC16F690 с 20 выводами, 18 из которых можно задействовать под ввод-вывод. Альтернативное решение рассматривается в следующем разделе.

## Распределенные вычисления

Люди одновременно могут делать несколько дел. Например, сердце программиста равномерно бьется, в то время, как его пальцы стучат по



клавиатуре. Он может вводить информацию одной рукой, отпивая в то же время кофе из чашки, которая находится в другой руке.

Программировать одновременно выполняемые процессы на одном микроконтроллере возможно, однако сложно. Выходом из ситуации является распределение задач между двумя или большим числом микроконтроллеров. Каждый из них работает независимо, если не учитывать периодический обмен сигналами **квитирования**, сообщающими о характере выполняемых задач. Такая схема известна как **распределенные вычисления.**

Примером распределенных вычислений может быть игра портального робота в шашки. Микроконтроллер на главной раме управляет двигателями и взаимодействует с оператором. Второй процессор, расположенный на раме "X", отвечает за камеру, которая сканирует игровое поле для регистрации позиций шашек. Логика игры реализуется микроконтроллером, установленным на основной раме (рис. 1.7).

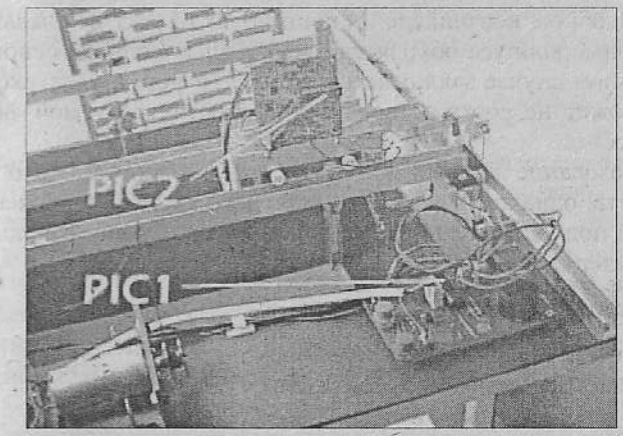



**Рис. 1.7.** Портальный робот (см. главу 10) использует два микроконтроллера для реализации более сложных операций: PIC1 — на основной, а PIC2 — на X-раме.

Еще одно преимущество такой системы заключается в том, что два микроконтроллера предоставляют больше каналов ввода-вывода, чем один. В случае портального робота наличие двух микроконтроллеров снижает число проводов, проложенных между основной и X-рамой.

# Глава 2. Механика роботов

## Материалы

Материалы для корпуса или рамы робота должны быть достаточно прочными, простыми в обработке, надежными и дешевыми. Кроме того, они должны хорошо выглядеть, т.е. иметь блестящую или привлекательно окрашенную поверхность.

Всеми этими качествами обладают некоторые виды пластмассовых пищевых контейнеров. Размещение робототехнических механизмов и схем в подобном контейнере иллюстрирует проект из главы 6, "Скутер". Подобные корпуса обходятся очень дешево, однако основная проблема в данном случае заключается в том, что форма пластиковых контейнеров может не соответствовать в точности требуемой форме или размерам робота.

Преобразование коробки для бутербродов в робота — это, конечно, быстрый путь, однако создание специального корпуса — более профессиональный подход. Некоторые из наиболее популярных используемых для этого материалов описаны в следующих подразделах.

### Алюминиевые изделия

Магазины с товарами для моделирования предлагают широкий ассортимент недорогих алюминиевых изделий различных размеров и поперечных сечений (рис. 2.1).

Алюминий также продается в листах. Его легко сверлить и резать ножовкой. Не слишком толстую полосу и профиль с квадратным сечением можно гнуть вручную. То же самое относится и к пруту. Преимущество алюминия, наверное, известно всем: это легкий метал. Легкие, но в то же время жесткие рамы можно перемещать с помощью двигателей малой

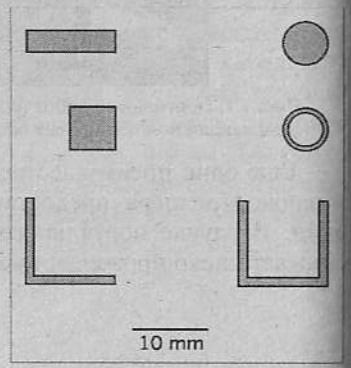



**Рис. 2.1.** Формы поперечных сечений алюминиевых изделий



мощности. Это, в свою очередь, означает, что для приведения их в действие необходимы маломощные батареи. Пример использования алюминиевой рамы рассматривается в главе 10, "Портальный робот". Этот проект был построен с помощью только двух видов профиля: полосы и желоба.

### Латунные изделия

Латунь пригодна для изготовления некоторых мелких элементов механизмов. Она продается в тех же магазинах с товарами для моделирования, однако в меньших по размеру заготовках. Латунь дороже алюминия, однако, к счастью, ее много не требуется. Она легко обрабатывается с помощью дрели и ножовки. Более тонкий профиль можно гнуть вручную.

Отличительная особенность латуни заключается в том, что она — достаточна жесткая и, в то же время, пружинит (чего не скажешь об алюминии). Именно поэтому латунь и подобные ей сплавы используют для изготовления электрических контактов и различных видов пружинных зажимов.

### Пластик

Магазины с товарами для моделирования предлагают широкий ассортимент пластмассы в виде прута, трубки, уголка, желоба и листа. Все такие заготовки — небольших размеров, однако они очень полезны. Обычно они изготовлены из ударопрочного полистирола, и для изготовления из него коробок и рам необходимо использовать специальные клеящие вещества.

Пластиковые изделия можно также найти в обычных хозяйственных магазинах. В частности, отдел сантехники предлагает широкий ассортимент трубок и других водопроводных компонентов, применимых при создании роботов. Например, пластиковые колпачки можно использовать в качестве световых щитков для инфракрасных датчиков.

Отдел садоводства предоставляет удобные пластмассовые трубки. Например, распорки, отделяющие секции робота "Искатель", вырезаны из длинного поливинилхлоридного (ПВХ) водяного шланга. Так, один из наиболее популярных для изготовления роботов материал — ПВХ в листах толщиной 3 мм.

В отличие от мягкого и хрупкого пенопласта, листовой ПВХ — твердый, однако обладает определенной сжимаемостью. Это означает, что гайки и головки болтов при зажатии вдавливаются на доли миллиметров в поверхность. Это снижает вероятность их ослабления в резуль-

тате вибрации. Лист ПВХ легко обрабатывается и режется. Все, что необходимо для вырезания из него фрагментов с прямыми краями, — стальная линейка и острый ремесленный нож. В добавок ко всему, такие листы изготовляют в обширной цветовой гамме. Так, например, при изготовлении робота "Искатель" использовался лист ПВХ яркого томатного цвета.

Аналогичный материал — пенополистирол, представляющий собой лист твердого пенопласта толщиной 5 мм, покрытый с обеих сторон пластиковой пленкой (с одной стороны — белая, с другой — цветная). Такой лист менее прочный, чем ПВХ, однако он так же удобен в обработке и пригоден для небольших, легких роботов типа "Скутер" или "Андроид". Корпус робота и другие его конструкции можно собрать с помощью клея.

### Дерево

Дерево редко воспринимается как материал, используемый в робототехнике, однако иногда оно очень даже полезно. Для своего веса дерево достаточно прочное, а также его легко резать, сверлить, окрашивать и клеить.

Поскольку различные строительные и столярные пиломатериалы обычно слишком велики для создания роботов, магазины товаров для моделирования предлагают очень легкую пробковую древесину, известную как **бальза**. Благодаря своей малой плотности и простоте обработки, она стала излюбленным материалом авиамоделистов. По этим же причинам бальза пригодна и в робототехнике, что подтверждает робот "Андроид".

## Крепеж

В большинстве случаев компоненты робота скрепляют с помощью болтов и гаек. В общем случае наиболее применимы болты с диаметром сечения 3 мм (М3), однако иногда требуются и размеры поменьше. Так, например, монтажные отверстия двигателя могут соответствовать болтам М2,5 или М2, а микропереключатели крепятся болтами с сечением 2 мм. Кроме того, держите про запас болты разной длины. Как правило, бывает достаточно 10–15 мм, однако иногда могут понадобиться и более длинные (до 25 мм) и короткие (до 6 мм) болты.

Позаботьтесь также о шайбах, которые выполняют несколько различных функций. Плоские шайбы распределяют нагрузку под головкой болта. Они полезны при соединении с помощью болтов относительно массивных элементов (например, двигателей) и относительно гибкой



пластины. Вибростойкие и пружинные шайбы предотвращают ослабление гаек. Используйте их для болтовых соединений двух металлических деталей. При крепеже к листовому ПВХ в них нет необходимости, поскольку этот материал сам по себе достаточно эластичен.

Нейлоновые шайбы и болты, поставляемые производителями электронных компонентов, необходимы в тех случаях, когда существует риск короткого замыкания. Это может случиться, если печатная плата крепится болтами к металлической панели. В таких случаях используют нейлоновые болты и шайбы или пластмассовые стойки.

Прокладки изготовляются в виде коротких трубок длиной 6..38 мм из металла или нейлона. Они предназначены для удержания печатной платы на небольшом расстоянии от панели, на которой она смонтирована, однако могут применяться и иначе. Короткие распорки мы еще иногда называем втулками.

Что касается клеев, то их существует великое множество, однако мы будем использовать только две разновидности: для обычной фиксации подойдет любой универсальный клей, применимый к пенопласту, а для быстрого, прочного склеивания — анаэробный суперклей. Капля такого клея, нанесенная на болтовое соединение, проникает в пространство между болтом и гайкой, после чего застывает. Это фиксирует гайку на болте, предотвращая ослабление соединения в процессе работы. При сборке роботов из металлических частей подобные клеи просто бесценны. Хотя они прочно удерживают гайку на болте, небольшого усилия гаечного ключа будет достаточно, чтобы в случае необходимости ослабить это соединение.

Представленный на рис. 2.2 клеевой пистолет следовало бы отнести к рабочему инструменту, однако более логично описать его вместе с клеями. Он плавит клеевые брикеты и через форсунку наносит расплавленный клей на склеиваемые поверхности. Клеевой пистолет — очень удобный инструмент, который при склеивании деталей всегда следует держать поду рукой.

Липкая лента, казалось бы, имеет мало общего с роботами, но фактически ее можно очень удачно использовать при фиксации деталей, которые не удается зафиксировать гайками, болтами или клеем. Типичные держатели батарей AAA и AA не имеют никаких монтажных отверстий, при этом их фактически негде просверлить, чтобы вставить болт. Именно для фиксации таких держателей и других подобных компонентов мы и будем использовать липкую ленту.

Наконец, всегда держите под рукой коробку канцелярских кнопок и пакет или рулон двухсторонней самоклеющейся ленты.

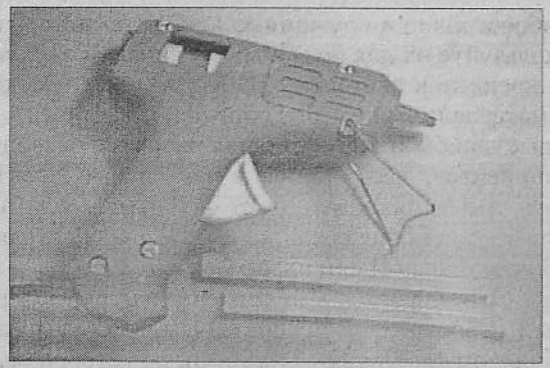


Рис. 2.2. Клеевой брикет топится электрической нагревающей обмоткой внутри пистолета. Для выдавливания расплавленного клея через форсунку следует нажать курок

## Инструменты

Инструменты, которые необходимы при сборке роботов, частично определяются используемыми материалами. Так, для работы с пенополистиролом основные инструменты — стальная линейка, ремесленный нож и пластмассовая разделочная доска (используйте ту, которая отслужила свой срок на кухне) или резиновый коврик. При монтаже двигателей и печатных плат потребуются также некоторые другие инструменты. Также, для конструирования роботов с алюминиевыми рамами (например, портальных), крайне важны вертикальный сверлильный станок и ножовка.

### Режущие инструменты

Малая ножовка с длиной лезвия 150 мм хороша для работ, наподобие резки древесины, пластмассы и печатных плат. Для резки алюминия или латуни более эффективна обычная ножовка по металлу. Если возникают трудности с вырезкой прямых или любых других точных углов, очень пригодиться станок для резки под углом (рис. 2.3). Он удерживает лезвие ножовки вертикально и перпендикулярно к поверхности материала и оснащен градуировкой, помогающей делать вырезы равной длины. Удерживающая лезвие рама может поворачиваться под углами, отличными от 90°. Для установки угла резки служит градуированная шкала. При конструировании портальных роботов без такого станка просто не обойтись.

Для закругления граней среза используйте мелкий напильник среднего размера. Набор надфилей удобен для расширения отверстий и придания формы небольшим деталям. Для формирования больших отверстий и многих других задач удобен грубый напильник с круглым сечением диаметром около 3 мм (рис. 2.4). Он быстро счищает материал и подходит для обработки металла, древесины и пластмассы.



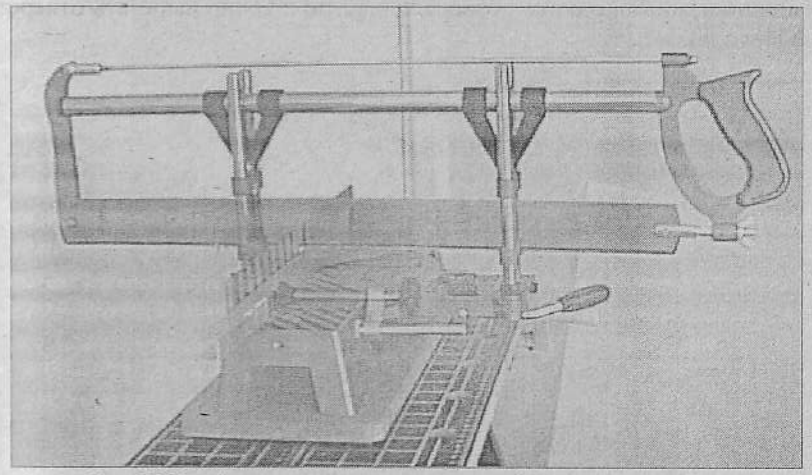


Рис. 2.3. Такой станок позволяет резать под фиксированным углом

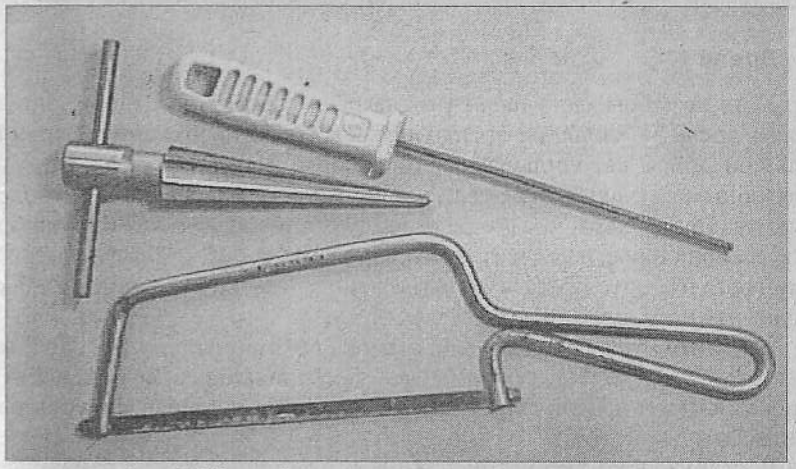


Рис. 2.4. Круглый напильник, развертка и малая ножовка

Для расширения круглых отверстий до 18 мм в диаметре служит развертка (см. рис. 2.4). Этот инструмент не относится к жизненно важным, однако справляется со своей работой превосходно. Если необхо-

димо вырезать большие отверстия, то можно воспользоваться циркулярной режущей насадкой электродрели (рис. 2.5). Она поставляется с набором сменных круговых режущих полотен для создания отверстий диаметром от 25 до 53 мм. Опять-таки, этот инструмент не является жизненно важным, однако позволяет быстро и точно вырезать отверстие большого диаметра.

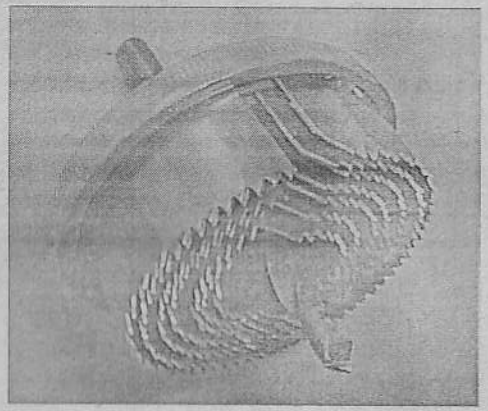


**Рис. 2.5.** Лезвия циркулярной режущей насадки прорезают отверстия различных диаметров, отцентрированных по отверстию, первоначально созданному сверлом

### Дрели

Хотя во многих случаях ручная дрель вполне приемлема, электрическая дрель — большое преимущество, когда необходимо выполнить большой объем сверлильных работ. Сборка робота, как раз, относится к подобным случаям. Даже если у вас уже есть маленькая электродрель для домашних работ, все же рекомендую приобрести что-нибудь более профессиональное. Так, вертикальный сверлильный станок недорог, однако гораздо более прост в использовании, чем электродрель, и обеспечивает намного лучшие результаты.

Вертикальный сверлильный станок, показанный на рис. 2.6, оснащен передачей ременного привода, обеспечивающей возможность изменения скорости. При сверлении в пластмассе лучше всего установить минимальную скорость.

У станка присутствует защитное ограждение, предохраняющее от ранения глаз вылетающими частицами материала, однако ношение защитных очков при работе с этим станком — важная мера предосторожности (особенно при сверлении в металле) (рис. 2.7). Еще одна важная мера безопасности — пара защитных перчаток.



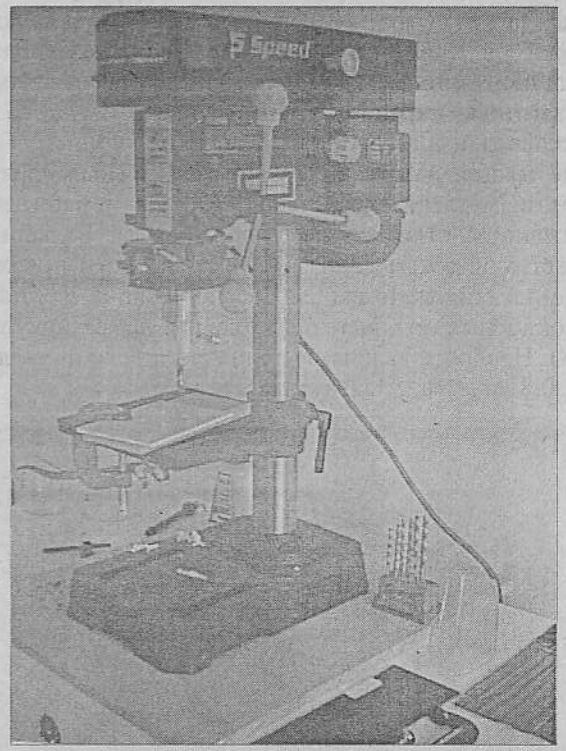



**Рис. 2.6.** Вертикальный сверлильный станок помогает точно позиционировать отверстия. Используйте его для сверления в алюминии, латуни, древесине или пластмассе (лучше — на самой низкой скорости)

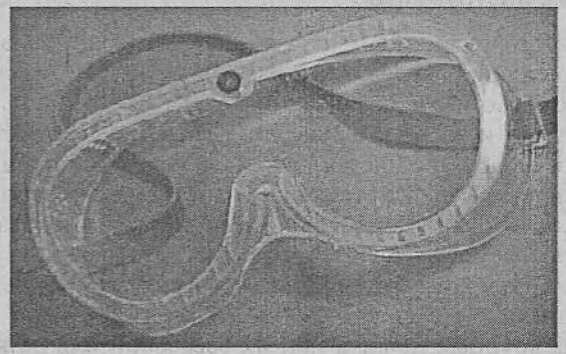


**Рис. 2.7.** Пластиковые защитные очки сохраняют глаза во время сверлильных работ

Для фиксации заготовки во время сверления очень удобен набор небольших зажимов. Кроме того, необходим набор сверл диаметрами от 1 до 6 мм. Для нанесения метки в точке сверления служат керн и молоток. Если на заготовку заранее не нанести такую метку, то сверло, скорее всего, соскользнет, и отверстие окажется смещенным.

Для более точной работы необходима маленькая дрель, управляемая низковольтным электродвигателем. Ее можно запитать от низковольтного источника питания монтажного стола. Большинство таких дрелей работает в широком диапазоне постоянных напряжений от 6 до 15 В. В их комплект обычно входят сверла, мини-развертки, проволочные щетки, войлочные полировочные диски и даже циркулярная мини-пила (рис. 2.8). Наиболее важными принадлежностями являются сверла диаметром от 0,8 до 2 мм, а также развертки.

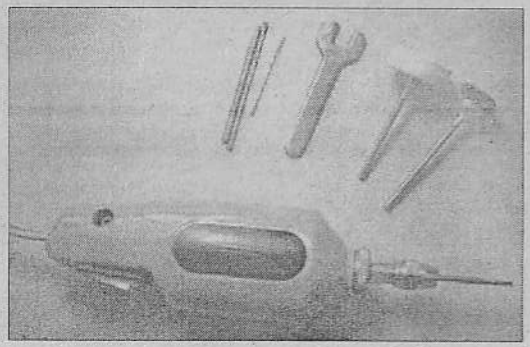


**Рис. 2.8.** Эта миниатюрная электродрель поставляется с большим набором насадок

### Зажимные инструменты

К этой категории инструментов относится пара стандартных плоскогубцев, пара длинногубцев и два разводных ключа, рассчитанных на максимальное развод 12 и 36 мм.

Для поднятия и удержания мелких деталей существует множество инструментов, однако в общем случае наиболее удобными из них оказываются пинцеты. Приобретите пару достаточно жестких пинцетов (ширина зубьев — около 2 мм) для удержания маленьких гаек и болтов, изгибания провода, извлечения микросхем из разъемов и множества других задач. Пинцет мы будем использовать гораздо чаще любых других инструментов.

Для более крупных деталей при их сверлении или резке с успехом используются переносные тиски. Самые дешевые тиски сделаны из пла-



стмассы и оснащены присосками на основании для временного закрепления на поверхности верстака.

Также при работе может пригодиться увеличительное стекло с крепежом (рис. 2.9) — особенно, если в процессе пайки необходимо удерживать небольшие печатные платы и электронные компоненты, наподобие разъемов и переключателей.

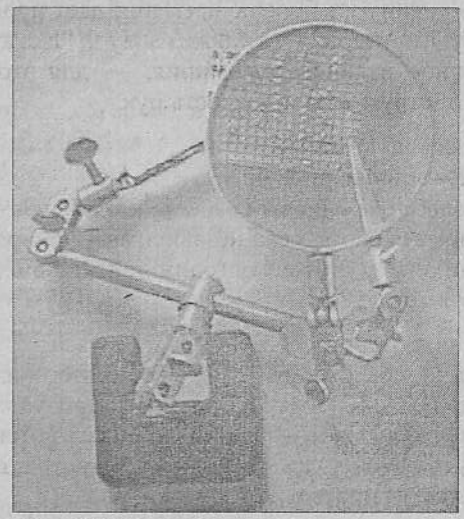


**Рис. 2.9.** Увеличительное стекло с крепежом

### Другие инструменты

Отвертки, очевидно необходимы, но только — малых размеров с обычными лезвиями шириной до 4 мм. При работе с метрическими болтами обычно бывает полезен накидной ключ соответствующего размера. Иногда может пригодиться также набор отверток для часовых механизмов. Набор маленьких напильников различных форм будет удобен для сглаживания граней срезов и придания формы деталям механизмов. Стоит также обзавестись набором надфилей. В качестве последнего элемента инструментария назовем стальную инженерную линейку длиной 300 мм.

## Проектирование корпуса мобильного робота

Перед началом сборки робота следует обдумать ряд аспектов. Первый из них — вариант шасси:

- какая-нибудь заготовка (например в виде пластмассовой коробки для завтраков) — такой робот просто и быстро собирается (см. главу 6.1, "Скутер");
- готовая игрушка — обычно все идет быстро и гладко, однако иногда возникают затруднения (см. главу 8, "Робот-игрушка");
- шасси, сделанное из листа пластмассы или пенополистиирола, — просто, однако отнимает больше времени; дает простор для творчества (см. главу 7, "Андроид", а также главу 9, "Искатель").
- шасси, сделанное из листа алюминия, — для этого потребуются специальные инструменты и ловкость рук.

Вопросы относительно колес:

- Сколько? Три — популярное число.
- Приводные колеса размещены спереди или сзади?
- Управление по танковому (два независимых приводных колеса, два двигателя) или автомобильному (один двигатель с дифференциальной передачей) методу?

Поверхность, по которой робот будет двигаться:

- твердая, плоская поверхность типа деревянного или выложенного плиткой пола, бетонного или кирпичного тротуара (для хорошего сцепления с ней могут потребоваться шины с протектором);
- лужайка — требуются колеса большого диаметра, обеспечивающие хороший дорожный просвет;
- наклонная поверхность — делайте робот широким и приземистым, чтобы он не перевернулся;
- ковровое покрытие — монтируйте колеса на шасси низко, чтобы между корпусом и ковром присутствовал просвет. Ковры часто создают трудности для роботов — особенно если они с бахромой по краям. Для более эффективного их преодоления следует использовать колеса большого диаметра. Ступени и лестницы — реальная проблема, для решения которой вместо колес придется задействовать гусеницы или даже научить робота ходить на ногах.

Размеры:

- небольшой размер (до 200 мм во всех направлениях) хорош для роботов, перемещающихся дома (им проще находить путь между мебелью);
- небольшие размеры могут не обеспечить достаточно пространства для батарей, двигателей, всех датчиков и исполнительных механизмов;



- из-за небольших размеров может быть затруднен доступ к схемам для их тестирования, а также — вставка и извлечение микроконтроллера PIC;
- большие размеры ведут к увеличению веса, необходимости в более прочном шасси, в большем количестве электроэнергии для приведения робота в движение, в более мощном (а следовательно, и более тяжелом) двигателе, в более мощных (больших и тяжелых) батареях. По возможности придерживайтесь минимализма!

Форма:

- роботу, размеры которого в длину и в ширину примерно равны, проще разворачиваться в ограниченном пространстве (например, в обстановке комнаты);
- непропорционально высокий робот, скорее всего, будет падать на неровностях, при столкновении с препятствием и при резком изменении скорости движения;
- если робот оснащен захватами для поднятия груза, то у него должно быть широкое основание для обеспечения устойчивости.

Более подробно вопросы проектирования рассматриваются в следующих двух разделах.

## Колеса

### Ходовые колеса

Основные параметры несущих колес — диаметр и протектор. Большой диаметр предпочтителен в случае грубой или неровной поверхности, поскольку колесу проще преодолевать бугры и менее вероятно, что оно застрянет в яме. Кроме того, большее колесо обеспечивает хороший просвет между поверхностью и дном корпуса.

Если поверхность гладкая и ровная (например, в случае рельсы портального робота), то преимущество отдается маленьким колесам, поскольку они меньше весят. Не будем забывать, что чем меньше вес робота, тем проще двигателю сдвинуть его с места.

Шины (рис. 2.10) помогают роботу перемещаться без проскальзывания. Просто запрограммированные роботы обычно начинают движение и останавливаются резко, что ведет к проскальзыванию или пробуксовке. Мы ожидаем, что робот будет перемещаться по прямой, однако проскальзывание колес приводит к тому, что он фактически движется по несимметричной кривой. Если он не будет непрерывно сверять свое местоположение по фиксированным ориентирам, то скоро полностью

потеряет ориентацию. Колеса проскальзывают даже при наличии шин, но все же — не так сильно.

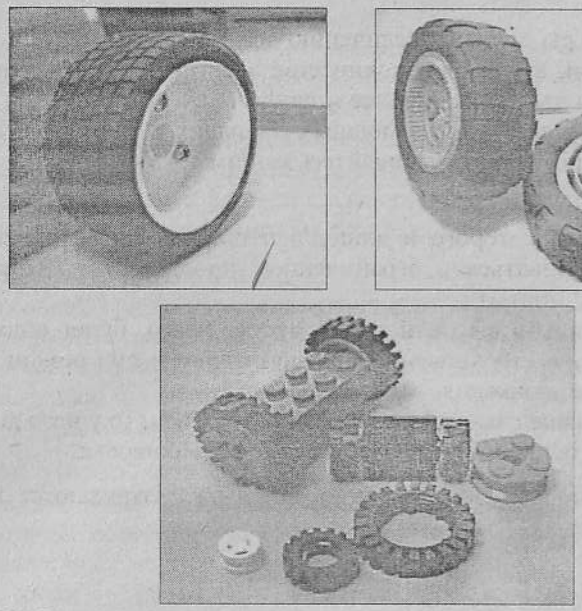


**Рис. 2.10.** Разнотипные шины, взятые из детских игрушек или приобретенные в магазине товаров для моделирования

Типичная проблема колес заключается в том, что диаметр отверстия в ступице зачастую не соответствует диаметру вала двигателя. Стандартов в данном случае не существует. Здесь потребуются эксперименты с различными сочетаниям колес и двигателей, а также — компромиссы и импровизация.

Колесо должно сидеть на валу плотно, иначе при приложении крутящего момента оно отвалится или будет проскальзывать. Обычно силы трения между ступицей колеса и валом двигателя вполне достаточно — особенно в случае легкого робота. Если диаметр вала меньше отверстия в ступице, то закрепите на конце вала отрезок трубки аквариумного аэратора или кусочек ПВХ изоляции (срезанной с кабеля), после чего плотно насадите колесо. Возможно, для того, чтобы соединение получилось прочным, потребуется второй слой прокладочного материала.

Если колесо садится достаточно плотно (не шатается при вращении), то проскальзывание можно устранить, прикрутив ступицу к валу.



### Зубчатые колеса

В трансмиссиях и для организации движения манипуляторов и схватов часто применяют зубчатые колеса. Они продаются в виде комплектов пластмассовых шестеренок различных диаметров, которые передают крутящее усилие за счет зацепления зубьев. Число зубьев на колесе важнее его диаметра, поэтому при обозначении шестеренки указывают именное это число (например, 24t или 36t).

Когда одна шестеренка сцепляется с другой, то скорость (число оборотов или угловая скорость) одного колеса относительно другого зависит только от числа зубьев на них. Предположим, двигатель вращает шестерню A с 10 зубьями, которая сцеплена с шестерней B с 40 зубьями. Шестерня A за полный оборот провернет шестерню B на 10 зубьев, что составляет для нее только четверть оборота. Сцепление с шестерней, имеющей меньшее количество зубьев, приводит к снижению скорости вращения.

В рассмотренном примере передаточное число определяется количеством зубьев на двух шестеренках. В данном случае этот коэффициент определяют как 40 к 10, или 4 к 1, что обычно записывается как 4:1. Электродвигатели обычно вращаются с высокой скоростью (несколько тысяч оборотов в минуту). Так, при вращении шестерни A из нашего примера со скоростью, скажем, 16 000 об/мин. шестерня B будет вращаться с 1/4 этой скорости, т.е. со скоростью 4 000 об/мин. Эта скорость определяется уравнением:

Скорость B = Скорость A × (Число зубьев A / Число зубьев B).

Значение 4 000 об/мин. — слишком большое для ходовых колес робота. Скорость была бы меньше, если бы на шестерне A было меньше, а на шестерне B — больше зубьев, однако размеры для этих двух шестеренок имеют пределы. По этой причине мы установим на один вал с колесом B третью шестеренку C, чтобы они вращались вместе. У этой шестерни — меньше зубьев, чем у B (например, 10), и она сцеплена с четвертой шестерней D, имеющей, например, 40 зубьев.

Для этого случая передаточные коэффициенты будут равны:

- от A к B — 4:1;
- от B к C — 1:1 (на одном валу);
- от C к D — 4:1.

Общее передаточное число от A к D будет равно 4 × 4 = 16:1. Если двигатель вращает шестерню A со скоростью 16 000 об/мин., то шестерня D будет вращаться со скоростью 16 000 / 16 = 1 000 об/мин.

Такую организацию шестеренок называют **зубчатой передачей**. Для обеспечения скорости вращения, приемлемой для ходовых колес, описанная выше передача нуждается еще в нескольких шестеренках, однако принцип зацепления шестерен с разным числом зубьев остается тем же. Простейшую коробку передач можно собрать из двух алюминиевых или ПВХ пластин, скрепленных болтами (обычно — в углах).

Некоторые практические примеры использования шестеренок и зубчатых передач показаны на рис. 2.11 – 2.14.

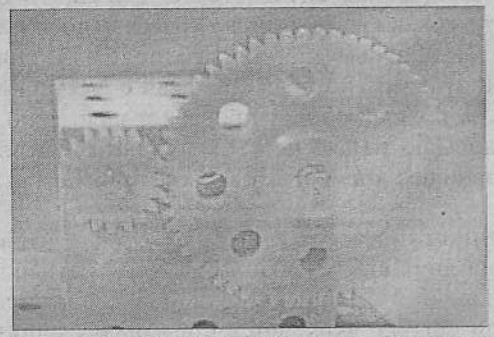


**Рис. 2.11.** У малой шестерни — 10 зубьев, у большой — 57. Передаточное число 5,7:1, поэтому большая шестерня вращается примерно с одной шестой скорости меньшей шестерни. Крутящий момент увеличивается примерно в 6 раз

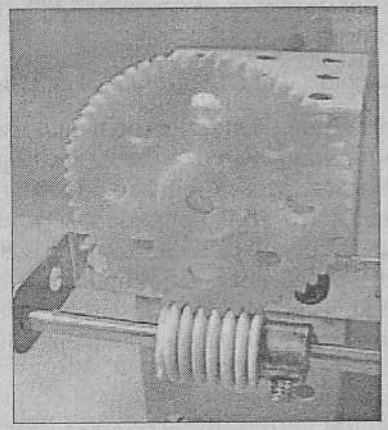


**Рис. 2.12.** Червячная передача. Один поворот червяка поворачивает шестерню с 58 зубьями на один зуб. Передаточное число составляет 57:1. Этот тип зубчатой передачи обеспечивает большой крутящий момент. Обратите внимание, что она — "однонаправленная", поскольку невозможно повернуть червяк, поворачивая большую шестерню. Шестерни установлены под прямым углом



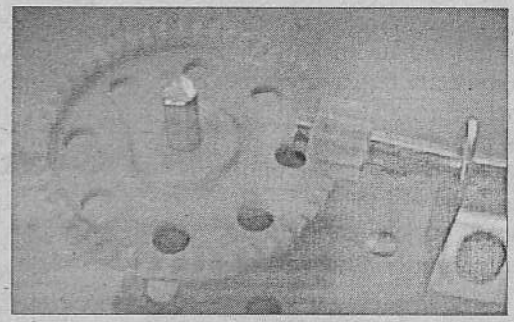


**Рис. 2.13.** Коронная и ведущая шестерни — пример понижающей зубчатой передачи с размещением валов под прямым углом. Ведущая шестерня имеет десять зубьев, а ведомая — 50. Передаточное число составляет 1:5, поэтому меньшая шестерня будет вращаться в пять раз быстрее большой

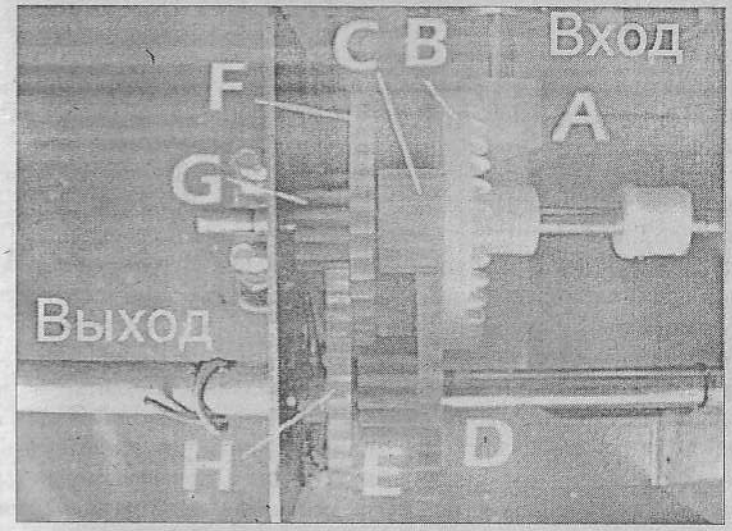



**Рис. 2.14.** Двигатель вращает входной вал, на который насажена шестерня A, сцепленная с коронной шестерней B. Маленькая передаточная шестерня C совмещена с шестерней B, чтобы вращаться вместе с ней. Она сцеплена с большей шестерней D, совмещенной с ведущей шестерней E. Узел D-E свободно вращается на валу. Шестерня E сцеплена с большой шестерней F, совмещенной с ведущей шестерней G. Узел G-H свободно вращается на валу. Шестерня G сцеплена с большой шестерней H, которая жестко посажена на выходной вал.
Количество зубьев: A = 10; B = 36; C = 14; D = 36; E = 14; F = 36; G = 14; H = 36.
Передаточное число от A к B составляет 1:3,6, а остальные три — 36:14 или 2,57:1. Общее передаточное число составляет:
1 : (3,6 × 2,57 × 2,57 × 2,57) = 1:61.

Понижающая зубчатая передача уменьшает скорость вращения и увеличивает **крутящий момент** (сила кручения вала). Значение крутящего момента зависит от приложенной силы и расстояния от центра вращения. Например, крутящий момент приводных двигателей робота "Искатель" (см. главу 9) равен 2,1 кгс*см. Это соответствует крутящему усилию, формируемому силой, эквивалентной 2,1 кг, действующей на колесо радиусом 1 см. Сила, равная половине указанной (1,05 кгс), действующая на удвоенном расстоянии (2 см), даст тот же самый крутящий момент (рис. 2.15).

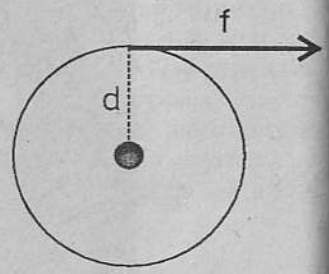



**Рис. 2.15.** Крутящий момент, произведенный силой f, действующей на расстоянии d, равен T = fd

Предположим, схват робота удерживает груз 0,2 кг, а длина его рычага составляет 15 см. Рычаг вращается, чтобы поднять груз. Требуемый крутящий момент равен 0,2 × 15 = 0,75 кгс*см. Двигатель, развивающий крутящий момент не более 0,5 кгс*см, при попытке поднять этот груз просто остановится.

Секрет успеха — в использовании понижающей зубчатой передачи. Так, двигатели робота "Искатель" оснащены коробкой передач с коэффициентом 82:1. Это снижает число оборотов выходного вала до 1/82 от числа оборотов вала двигателя, т.е. до 70 об/мин. В то же время, крутящий момент увеличивается в 82 раза относительно крутящего момента на валу двигателя (практически — немного меньше из-за трения).

Крутящий момент, указываемый для двигателей, зависит от рабочего напряжения. Чем оно ниже, тем ниже и реальный крутящий момент.

Большинство применений двигателей требует использования понижающей зубчатой передачи, однако аналогичный результат иногда можно получить и другими способами. Хорошим пример этого — механизм рулевого управления, используемый в роботе-игрушке (глава 8). Ремень одет непосредственно на выходной вал. Диаметр шкива в 12,5 раз превосходит диаметр этого вала, что дает понижающее передаточное число 1:12,5.

### Шкивы

Шкив — это колесо с канавкой по ободу. Шкивы, главным образом, используются для передачи усилия. Так, например, в портальном роботе они передают силу тяжести на шасси, чтобы перемещать его по рельсам по мере раскручивания лебедки. Они также используются в блочных



системах, поднимающих и опускающих крюки и некоторые другие инструменты. В данном случае сила передается с помощью ремня между двумя шкивами.

Пара шкивов одинакового диаметра просто передает силу на расстояние. Если их диаметры различны, то результат аналогичен применению зубчатой передачи. Большой шкив, приводимый в движение меньшим шкивом, вращается медленнее, однако с увеличением крутящего момента (рис. 2.16 – 2.18).

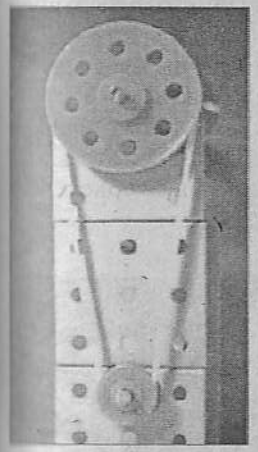


**Рис. 2.16.** Шкивы передают энергию на расстояние. Если они — разных диаметров, то будут вращаться с разной скоростью. Диаметры канавок в данном примере — 36 мм и 12 мм. Получаем 36/12 = 3, т.е. один оборот большего шкива соответствует трем оборотам меньшего

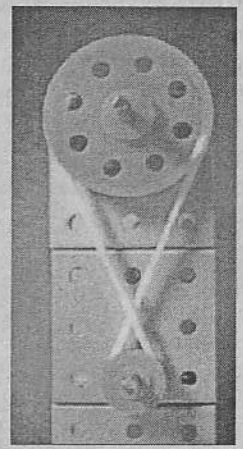


**Рис. 2.17.** Реверсирование направления вращения за счет перекрещивания приводного ремня

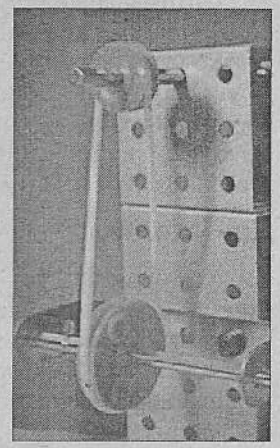


**Рис. 2.18.** Валы могут устанавливаться под прямым или под любым другим острым углом. При этом следует помнить: если шкивы разместить слишком близко друг к другу, то существует риск соскакивания с них передаточного ремня

В отличие от шестеренок, которые должны быть в контакте, шкивы связывают ремнем, что позволяет передавать силу на расстояние. Другое преимущество шкивов заключается в том, что приводные ремни немного эластичны, поэтому точное расстояние между шкивами не существенно. В случае же зубчатых передач очень важно, чтобы зубья двух шестеренок сцеплялись корректно.

Недостаток шкивов заключается в том, что ремень может соскочить или порваться. Другая проблема состоит в проскальзывании ремня, если груз — слишком тяжелый, но это — не всегда плохо. Ремень вносит в систему **эластичность**. Например, пальцы схвата замыкаются на объекте, который должен быть надежно удержан. Если захват окажется слабым, то груз может выпасть, если же он будет слишком сильным, то груз может быть поврежден. Точное положение пальцев при захвате объекта определить очень сложно.

Оптимальный подход в этом случае — ввести в систему небольшую эластичность. Один из вариантов заключается в использовании схвата только для подъема упругих объектов, наподобие резиновых мячей. Другой путь, позволяющий работать с любыми предметами, — задействовать в механизме упругий передаточный ремень. Как только пальцы схвата замкнутся на объекте, ремень начинает скользить. Двигатель при этом продолжает работать, формируя усилие, удерживающее объект.

## Двигатели

Хотя двигатели — электрические устройства и, следовательно, могли бы рассматриваться в главе, посвященной электронике, они приводят в движение механические детали, поэтому мы поговорим о них здесь. В центре нашего внимания будут небольшие низковольтные двигатели постоянного тока (рис. 2.19).

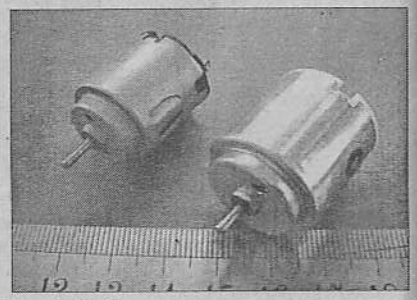


**Рис. 2.19.** Пара очень дешевых двигателей на 1,5 В и 4,5 В. Они формируют крутящий момент 8 гс*см и 18 гс*см соответственно. Эти двигатели не имеют коробки передач

При выборе двигателя для проекта главный аспект — его рабочее напряжение, поскольку оно частично определяет размеры и вес портативной батареи. Двигатели на 12 В требуют восьми батареек AA, что слишком много для маленького робота типа "Скутер". Робот "Искатель" больше и тяжелее, поэтому для его перемещения требуются более мощные двигатели на 12 В, питаемые от блока из восьми элементов AA. Для достижения максимальной мощности следует использовать сухие батареи, однако аккумуляторы более экономичны. Восемь LiMH-элементов дают напряжение 9,6 В, чего достаточно для приведения в действие двигателей.



Следующий важный аспект — коробка передач. Механизмы робота очень редко приводятся в действие напрямую от двигателя, вал которого вращается со скоростью в несколько тысяч оборотов в минуту. Если двигатель не оснащен встроенной коробкой передач, то почти наверняка ее придется собрать самому или купить готовую (рис. 2.20). Встроенная коробка передач — более элегантное, удобное и, пожалуй, дешевое решение (рис. 2.21).

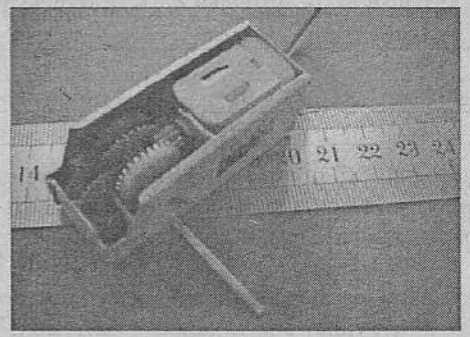


**Рис. 2.20.** Двигатель на напряжение от 1,5 В до 3 В с подсоединенной металлической коробкой передач. Вариант недорогой, однако относительно шумный

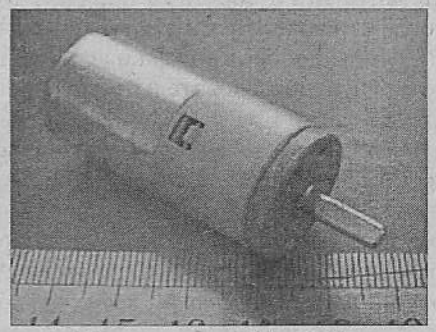


**Рис. 2.21.** Электромотор на 12 В со встроенной коробкой передач. Работает очень тихо. Выходной вал — диаметром 4 мм, с плоским концом для более надежного крепления

Наконец, важен вопрос крепления выходного вала с приводимым в действие механизмом. О проблемах, которая при этом часто возникают, уже упоминалось ранее. По возможности выбирайте двигатель с валом, который соответствует стыковочному узлу. Обычно хорошо подходит вал диаметром 4 мм. Для стыковки более толстых валов продаются специальные переходники, однако они относительно дороги, громоздки и увеличивают вес робота, чего следует избегать.

### Шаговые двигатели

Типичный шаговый двигатель содержит четыре набора катушек, расположенных таким образом, чтобы ротор переходил из одного положения в другое по мере запитывания катушек в фиксированной последовательности. Эта последовательность показана в табл. 2.1.

**Таблица 2.1.** Последовательность импульсов для управления шаговым двигателем

| Номер шага | Катушка 1 | Катушка 2 | Катушка 3 | Катушка 4 |
|---|---|---|---|---|
| 0 | Вкл. | Выкл. | Вкл. | Выкл. |
| 1 | Выкл. | Вкл. | Вкл. | Выкл. |
| 2 | Выкл. | Вкл. | Выкл. | Вкл. |
| 3 | Вкл. | Выкл. | Выкл. | Вкл. |

Эта последовательность повторяется, и на любом шаге две катушки включены, и две — выключены. Последовательность импульсов для управления шаговым двигателем можно сформировать микроконтроллером.

При переходе от шага к шагу, вначале изменяют свое состояние катушки 1 и 2, затем — катушки 3 и 4, далее — катушки 1 и 2 и т.д. В результате ротор вращается по часовой стрелке на 15° за один шаг. Если последовательность запустить в обратном порядке, то ротор будет вращаться против часовой стрелки. Шестикратное прохождение последовательности дает один полный оборот ротора шагового двигателя.

Скорость прохождения последовательности переключений зависит от временных характеристик программы. Двигатель совершает один оборот на 24 импульса. Изменение частоты импульсов изменяет скорость вращения вала. На любом шаге ротор может быть задержан в фиксированной позиции путем остановки последовательности. Если шаговый двигатель используется для приведения в действие манипулятора робота, то этот манипулятор можно точно позиционировать, запрограммировав микроконтроллер на формирование требуемого числа импульсов. При этом отпадает необходимость в концевых выключателях, поскольку робот всегда будет знать положение его конечностей. Он сможет двигаться точно от одной позиции к другой, просто задавая количество генерируемых импульсов. Как только требуемая позиция достигнута, ротор блокируется и далее не вращается. Крутящий момент, требуемый для преодоления магнитного поля, удерживающего ротор в фиксированном положении, составляет несколько сотен грамм-сил на сантиметр.

Другое преимущество шагового двигателя заключается в возможности точного управления скоростью вращения, на которую не влияет



нагрузка (разве что, за исключением чрезмерной нагрузки, полностью останавливающей вал). Вероятность останова, перегрева и перегорания обмоток шагового двигателя меньше, чем для обычного двигателя.

Воспользовавшись немного модифицированной последовательностью переключений, можно заставить шаговый двигатель поворачивать ротор на 7,5° за шаг. Также выпускаются двигатели, обеспечивающие поворот 1,8° за шаг. Программируя шаговый двигатель, не подавайте на него импульсы слишком быстро, поскольку при чрезмерных скоростях возникает вероятность пропуска шагов, что ведет к ошибке позиционирования.

Казалось бы, шаговые двигатели — идеальный вариант для роботов, однако в проектах, описанных в этой книге, они не используются. Одна из причин — в сложности программной генерации требуемых последовательностей импульсов, когда микроконтроллер занят решением других задач. Кроме того, шаговые двигатели требуют большого тока и рабочего напряжения не менее 12 В.

### Серводвигатели

**Серводвигатель** (рис. 2.22) рассчитан на вывод вала в заданную угловую позицию. Он имеет три соединения со схемой управления. Два из них используются для подачи положительного и нулевого питающего напряжения, а по третьему передается сигнал управления (например, от микропроцессора).


**Рис. 2.22.** Небольшой серводвигатель, используемый в летающих моделях самолетов и в роботах. "Рога" (белые рычаги) используются для соединения двигателя с управляемым механизмом

Возможности вращения ротора серводвигателя ограничены. В общем случае он может поворачиваться на 60–90° в любую сторону от своего центрального положения.

Сигнал управления представляет собой последовательность импульсов, передаваемых с интервалом около 18 мс, т.е. с частотой 50 импульсов в секунду.

Угол поворота контролируется длительностью импульса:

- 1 мс — максимально возможный поворот влево;
- 1,5 мс — поворот в центральную позицию;
- 2 мс — максимально возможный поворот вправо.

Промежуточные значения длительности импульсов выводят ротор в промежуточные положения.

Серводвигатели часто используют в роботах, авиа- и автомоделях, поэтому их всегда можно купить в соответствующих магазинах.

## Соленоиды

Соленоид представляет собой продолговатую многовитковую обмотку из провода и сердечник из мягкого железа (рис. 2.23). Когда через обмотку протекает ток, сила магнитного поля затягивает сердечник в обмотку. При отключении тока ничего не произойдет, поэтому для извлечения сердечника из обмотки следует использовать пружинный механизм (или силу тяжести).

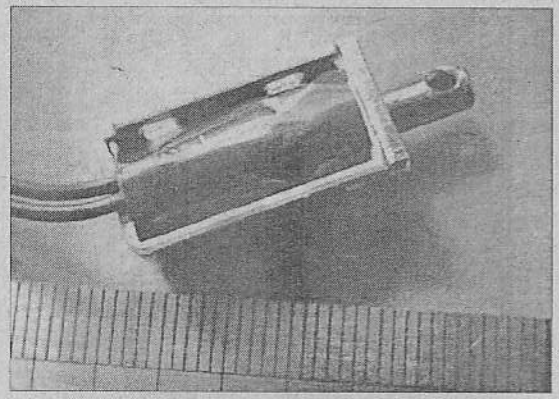


**Рис. 2.23.** Миниатюрный соленоид, пригодный для использования в роботах

Соленоиды используются для обеспечения линейного движения, чтобы *тянуть* что-либо по прямой линии. Если в своей начальной позиции сердечник достаточно глубоко утоплен в обмотку, то сила, действующая на него, может быть достаточно большой. Соленоид на 12 В может развить тянущее усилие, равнозначное одному-двум килограммам силы. Проблема заключается только в том, что ход сердечника (т.е. дистанция, которую он проходит) составляет всего лишь несколько миллиметров. Это ограничивает применение соленоидов в роботах, в основном, реализацией коротких перемещений в электрических замках, клапанах и т.п.



## Ссылки в Internet

Всемирная сеть — наилучшее средство наведения контактов с поставщиками, работающими по почтовым заказам. Перечислим некоторые полезные ссылки:

- `www.robotstore.com` — поставка широкого ассортимента деталей по почте;
- `www.legoshop.com` — поставка деталей Lego по почте;
- `www.tamiya.com` — описание изделий и список представителей компании Tamiya по всему миру;
- `www.brickline.com` — детали Lego;
- `www.robotbooks.com` — много интересных книг;
- `www.budgetrobotics.com`;
- `www.robohoo.com`;
- `www.hobbyengineering.com` — многие детали для роботов, компоненты и наборы.

# Глава 3. Электроника роботов

## Материалы

### Печатная плата

Компоненты электронной схемы почти всегда собирают на прямоугольной печатной плате, которую изготовляют из изолирующего материала. На ее обратной стороне проложены медные проводящие дорожки, формирующие соединения между компонентами. В схемах, описанных в этой книге, мы используем компоненты с проволочными выводами, которые пропускаются через отверстия в плате и припаиваются к проводящим дорожкам с другой стороны. Еще один тип электронных компонентов — для поверхностного монтажа (SMD). У них выводы — с площадками, которые припаиваются к дорожкам *на той же* стороне платы. Для их установки отверстия не требуются. Компоненты SMD обычно очень малы и с ними сложно работать, поэтому они в данной книге не используются. Будьте внимательны, чтобы по ошибке не купить компоненты для поверхностного монтажа.

В этой книге мы собираем схемы на платах с параллельными медными полосками (по десять на дюйм), перфорированных отверстиями (тоже по десять на дюйм) (рис. 3.1 и рис. 3.2).

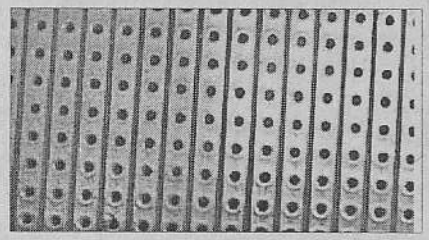


**Рис. 3.1.** Сторона медных проводников стандартной платы с полосками

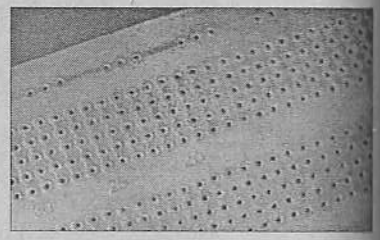


**Рис. 3.2.** Одна из более сложных печатных плат, предназначенная для установки микросхем (включая микроконтроллеры PIC)

Эти платы можно приобретать готовыми к использованию. Они позволяют варьировать монтаж в соответствии с размерами компонентов,



а также допускают внесение изменений позже, если в этом возникнет необходимость. Впрочем, более экономично приобретать такие платы в виде большого листа (приблизительно 300 мм в длину и 100 мм в ширину), а затем нарезать его на фрагменты малой ножовкой.

Компоненты соединяются посредством припаивания их к полоскам, а также — соединения некоторых полосок с помощью проводов, расположенных поперек платы под прямыми углами. Такую компоновку называют "манхэттенской" (рис. 3.3).

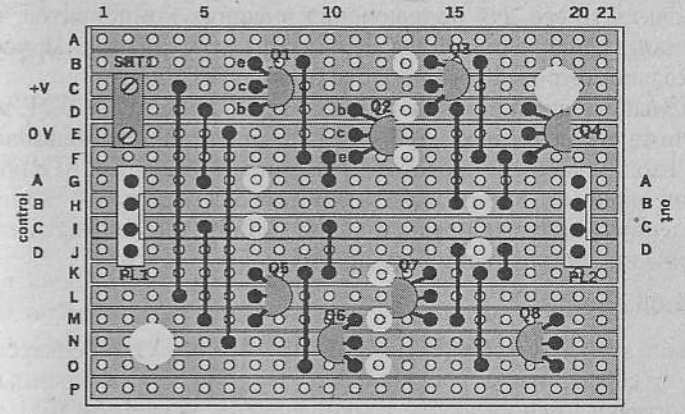



**Рис. 3.3.** Типичная компоновка платы с полосками, где полоски идут горизонтально, а соединительные проводники — вертикально

Раз уж мы вспомнили о компоновке печатных плат, то уместно будет сказать несколько слов о монтажных схемах, представленных в этой книге. Прежде всего, они нарисованы так, как будто плата — прозрачная, и показывают электронные компоненты на лицевой и проводящие полоски на обратной стороне. На них также показаны круглые вырезы в полосках, которые на стороне монтажа в действительности не видны. Два больших белых круга на плате — это отверстия, просверленные в печатной плате для крепления болтами.

Большими черными точками обозначаются места припаивания проводников к полоскам *под* платой. Они тоже со стороны монтажа не видны.

Среди электронных компонентов, показанных на рис. 3.3, находится клемма с двумя зажимными винтами (слева вверху), две монтажные колодки на четыре контакта (по центру слева и справа) и четыре транзистора.

### Соединительные провода

Может потребоваться три вида соединительных проводов.

- Одножильный изолированный провод диаметром 0,71 мм. Используется для монтажа на печатной плате. Продается на метры. Целесообразно всегда иметь под рукой один-два метра этого провода двух-трех цветов.
- Многожильный гибкий изолированный провод (иногда называемый звонковым) 13 × 0,12 мм (т.е. 13 жил диаметром 0,12 мм каждая). Используйте его для подключения внешних компонентов, наподобие двигателей, а также для межплатных соединений. Приобретите несколько метров такого провода нескольких цветов.
- Луженый медный провод без изоляции диаметром 0,71 мм. Для монтажа на печатных платах мы предпочитаем использовать его, а не изолированный провод. Отсутствие необходимости снятия изоляции экономит время, а риск короткого замыкания — минимален. Оголенные провода предоставляют множество контактных точек, которые можно использовать при тестировании схем.

### Припой

Припой — это смесь 60% олова и 40% свинца. Он продается в виде провода с сердечником из канифоли. При пайке электронных схем предпочтительно использовать тонкий провод диаметром 0,71 мм. Недавно были приняты законы, в соответствии с которыми в некоторых странах, включая ЕС, больше нельзя использовать свинцовый припой. Припой, не содержащий свинца, обычно представляет собой смесь олова и сурьмы. Он продается в виде провода диаметром 0,71 мм.

### Изоляция

Зачастую очень важно предотвратить протекание тока там, где его не должно быть. Для оборачивания "рискованных" проводов и выводов используйте ПВХ изоленту. Используйте ее также для изоляции точек спайки двух проводов. Будет вполне достаточно по одной катушке красной и голубой изоленты.

Изолирующие ПВХ трубки имеют несколько миллиметров в диаметре и выпускаются в широкой цветовой гамме. Они удобны для защиты оголенных проводов и мест спайки двух проводов. Большинство таких трубок опрессовывается нагревом. Если трубка одевается слишком свободно, то достаточно подержать около нее горячий паяльник, и она сожмется в диаметре наполовину. Это жестко закрепит изоляцию на месте.



Сжимающиеся под воздействием тепла трубки продаются упаковками с различными диаметрами и цветами. Одной такой упаковки хватает на несколько лет.

## Инструменты

Многие из инструментов, перечисленных в предыдущей главе, используются и при разработке электронных схем. К их числу относятся маленькие отвертки, длинногубцы, малая ножовка, напильник, пинцет, мини-дрель и лупа.

### Инструменты разработчика

Рассмотрим инструментарий, используемый на стадии проектирования до фактического монтажа схемы.

Блок питания монтажного стенда должен выдавать диапазон постоянных напряжений, до 12 В, регулируемых при токе до 1 А. Недорогой альтернативой, покрывающей большинство потребностей, является сетевой блок питания (рис. 3.4). Он напоминает обыкновенный сетевой штепсель и подключается непосредственно в розетку.

**Рис. 3.4.** Этот сетевой блок питания обеспечивает выходное постоянное напряжение 3; 4,5; 6; 7,5; 9 или 12 В. Максимальный ток — 1 А

Еще более простой источник питания — батарейный отсек с количеством элементов, необходимым для получения требуемого напряжения. При этом можно использовать батареи, которые будут установлены внутри завершенного робота. Новые щелочные элементы дают по 1,5 В каждый. Батареи NiCD и NiMH дают по 1,2 В, и их можно перезаряжать. Для таких элементов питания понадобится зарядное устройство, однако с точки зрений долгосрочных перспектив такой вариант — наиболее экономичный.

Перед пайкой на печатной плате схемы разрабатываются и тестируются на макете, который представляет собой массив контактных гнезд, объединенных в группы (рис. 3.5). Для создания электрических связей между этими гнездами служат провода с очищенными от изоляции концами.

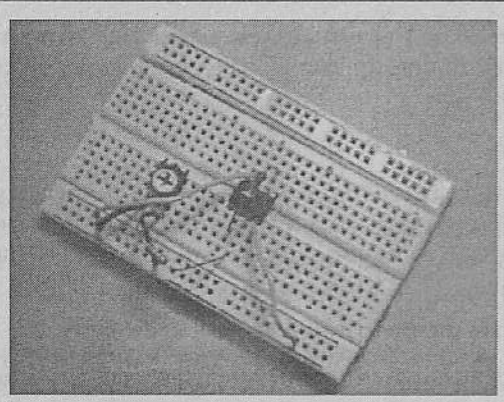


**Рис. 3.5.** Макетирование — первый практический этап разработки схем

При работе с макетом используют широкий ассортимент проволочных перемычек разной длины. Они продаются в наборах, однако их можно изготовить самостоятельно, нарезав нескольких десятков фрагментов *одножильного* провода и очистив от изоляции их концы. Они должны варьироваться по длине от 2 до 10 см.

Также рекомендуем заготовить несколько проводов, очищенных от изоляции на одном конце, и с миниатюрным зажимом типа "крокодил", припаянным на другом конце. Еще один полезный элемент при макетировании и общем тестирования схемы — набор разноцветных тестовых проводников. Их изготовляют из тонкого, гибкого провода с пружинным зажимом на обоих концах.

Еще один обязательный предмет — это тестер или мультиметр. Желательно, чтобы он был цифровым и измерял постоянные напряжения до 20 В. Он также должен измерять токи до 1 А, однако замеры токов в робототехнических схемах выполняют редко. Кроме того, измерительный прибор должен измерять сопротивление и емкость. Неплохо также, чтобы он реализовал функции проверки непрерывности цепи и тестирования диодов.

Цифровые измерительные приборы более предпочтительны из-за их высокого входного сопротивления (они берут очень мало тока от тестируемой схемы) и четырехзначной точности (для большинства целей вполне достаточно трех знаков). Однако быстрые колебания напряжения производят раздражающую и нечитабельную пляску цифр на индикаторе. Это именно тот случай, когда на сцену выходит аналоговый измерительный прибор. Подойдет даже дешевая старая модель, которую вы уже собирались выбросить. Его колеблющаяся стрелка расскажет о бросках напряжения почти все, что необходимо.



### Паяльный инструмент

В первую очередь необходим паяльник. Паяльная станция с термостатическим управлением — это, конечно, прекрасно, но будет вполне достаточно и простого паяльника с электрическим нагревом. Самое главное, чтобы он был маломощным (около 15 Вт), а его жало не должно превышать 2 мм в диаметре.

Если доступно несколько моделей паяльников, то выберите соответствующую вышеперечисленным требованиям, с маломощным шнуром питания. Некоторые паяльники "портит" мощный шнур, которым можно было бы запитать устройство мощностью 1 кВт. Толстый кабель затрудняет точную пайку. Еще одна важная принадлежность — подставка для паяльника. Она должна быть оснащена увлажняемой губкой для протирки паяльника.

Следующая паяльная принадлежность — теплоотвод. Он может выглядеть по-разному, однако суть остается прежней: теплоотвод — это кусок меди или алюминия, который крепится зажимом к выводу электронного компонента, когда он припаивается к плате. Его обычно используют при пайке полупроводниковых устройств, наподобие диодов и транзисторов. Теплоотвод крепится между выводом, который припаивается, и корпусом компонента. В результате тепло от места пайки идет через теплоотвод, а не электронный компонент.

### Другие инструменты для работы с электроникой

Для очистки изоляции с проводов и кабелей используют специальные щипцы. Это значительно экономит время, однако большинство таких инструментов рассчитано на работу с толстыми магистральными или антенными кабелями, а не с тонкими проводами, применяемыми в электронных проектах. Учтите это при покупке щипцов.

Для обрезки выводов компонентов после того, как они были припаяны, как нельзя лучше подходят бокорезы (косые острогубцы), а последним пунктом в нашем списке значится точечный нож (рис. 3.6). Он напоминает сверло с пластмассовой ручкой и используется для вырезки круглых отверстий в медных полосках. При прорезке печатных проводников ВСЕГДА исследуйте каждый разрез через лупу, поскольку зачастую по краям выреза остаются микроскопические медные волоски. Просто удивительно, как много тока может протечь через такую почти невидимую нить! Если схема не работает, как ожидалось, первым делом поищите подобные нити (конечно же, вооружившись лупой).

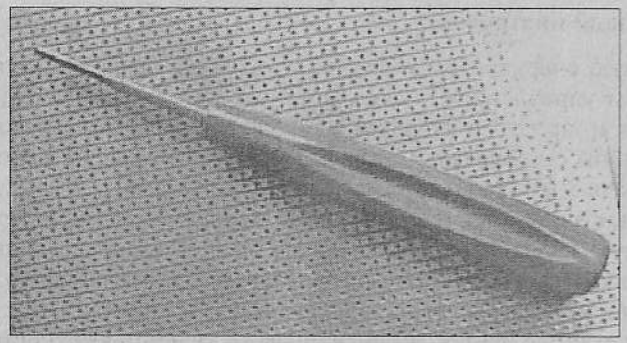


**Рис. 3.6.** Точечный нож для прорезки печатных проводников

## Электронные компоненты

При сборке одного из проектов, описанных в этой книге (как и любых других по готовым описаниям), покупайте только те электронные компоненты, которые перечислены, — не более. Впрочем, конечно же, несколько запасных транзисторов и других элементов никогда не помешают на случаи перегрева или непреднамеренных коротких замыканий.

С другой стороны, при разработке собственных схемы или модификации существующих, понадобится небольшой запас популярных электронных компонентов. Вот наши рекомендации…

- **Резисторы**. Необходимо по несколько штук каждого номинала из ряда E12 от 100 Ом до 1 мОм. Оптимальный вариант — металлопленочные резисторы с допуском 1% на 0,5 Вт. Они обычно продаются упаковками по 10 штук.

> ***Резисторы E12***
>
> Их номиналы: 10, 12, 15, 18, 22, 27, 33, 39, 47, 56, 68 и 82. Далее идут значения, полученные умножением основного ряда на множители: x10, x100, x1000 и т.д. до 10 МОм. Существуют также множители 0,1 и 0,01, но они используются редко.

- **Переменные резисторы**, устанавливаемые горизонтально, прекрасно подходят для макетирования. Удобные номиналы: 470 Ом; 1 кОм; 4,7 кОм; 10 кОм; 100 кОм и 1 МОм.
- **Конденсаторы**. Их основная задача в наших схемах — сглаживание всплесков и импульсов напряжения питания. Целесообразно обзавестись полиэфирными МКТ-конденсаторами на 100 нФ.
- **Транзисторы**. Чаще всего используются транзисторы BC548 в ключах, однако для этих целей применимы и другие типы (табл. 3.1) (рис. 3.7).



**Таблица 3.1.** Биполярные плоскостные транзисторы

| Тип | Проводимость | Макс. ток (мА) | Усиление по току* |
|---|---|---|---|
| BC327 | p-n-p | 800 | 100–600 |
| BC337 | n-p-n | 800 | 100–600 |
| BC548 | n-p-n | 100 | 110–800 |
| BC639 | n-p-n | 1000 | 200 |
| BC640 | p-n-p | 1000 | 200 |
| 2N3904 | n-p-n | 200 | 300 |

* Для тока коллектора 10 мА. Транзисторы часто различают по коэффициенту усиления. Так, например, у BC548C коэффициент усиления больше, чем у BC548A и BC548B.

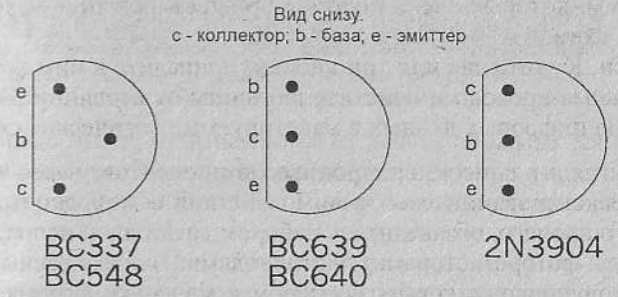



**Рис. 3.7.** Распределение выводов транзисторов

Когда источник сигнала неспособен обеспечить ток, достаточный для управления биполярным плоскостным транзистором, в качестве ключей и усилителей применяют полевые МОП-транзисторы (метал-оксид-полупроводник). В качестве ключей они могут управлять большими токами, и большинство типов таких транзисторов характеризуются очень низким сопротивлением состояния проводимости. Основной недостаток полевых МОП-транзисторов заключается в ощутимых вариациях порогового напряжения (т.е. напряжения на затворе, переводящее транзистор в состояние проводимости).

Полевой МОП-транзистор, представляющий интерес для маломощных схем, — это VN10KM (ток до 500 мА, сопротивление в состоянии проводимости 5 Ом, крутизна характеристики прямой передачи 0,2 S). Значение 1 S означает, что изменение напряжения на затворе на 1 В приводит к изменению тока на 1 А. Транзистор 2N7000 пропускает ток до 500 мА при сопротивлении состояния проводимости 5 Ом и крутизне характеристики прямой передачи 0,1 S. Его пороговое напряжение находится в диапазоне 1..2,5 В, а в типичном случае составляет 2 В (рис. 3.8).


**Рис. 3.8.** Распределение выводов двух маломощных полевых МОП-транзисторов

- **Светодиоды**. Приемлемы светодиоды стандартной яркости диаметром 5 мм, которые очень полезны при тестировании выходов логических схем.
- **Кнопки**. Купите две или три кнопки, припаяйте к ним короткие одножильные провода и очистите их концы от изоляции. Это обеспечит ввод цифровых данных в макетируемые логические схемы.

Так выглядит список электронных компонентов, часто используемых при макетировании схем взаимодействия с микроконтроллерами. Кроме всего прочего, обзаведитесь набором датчиков и исполнительных механизмов: фоторезисторами, фотодиодами, инфракрасными светодиодами, полупроводниковым зуммером и маленьким громкоговорителем. Конкретный перечень компонентов зависит от поставленных задач. В случае необходимости припаяйте к их выводам короткие проводники, чтобы они были готовы к подключению к макетной плате.

### Соединители

Эта книга основана на идее схемных модулей, которые можно собирать многими различными способами, формируя разные роботы. Модули также имеют то преимущество, что их монтажные платы малы и в силу этого легко размещаются в сжатом внутреннем пространстве робота. Модульную систему можно улучшать, дополнять и модифицировать без перестройки всей системы.

Для робототехнических схем приемлемы соединители нескольких типов. Самые дешевые соединители — это штырьковые выводы длиной 0,9 или 1 мм. Они пропускаются через отверстия в полосках монтажной платы и припаиваются. Конец соединительного провода изгибается в форме маленького крючка и припаивается к выводу. Паяные соединения очень надежны, однако создают трудности, если элементы необходимо заменить.



Штырьковые выводы печатных плат аналогичны обычным выводам, однако они длиннее и обычно позолочены (рис. 3.9). Они также пропускаются через отверстия и припаиваются.

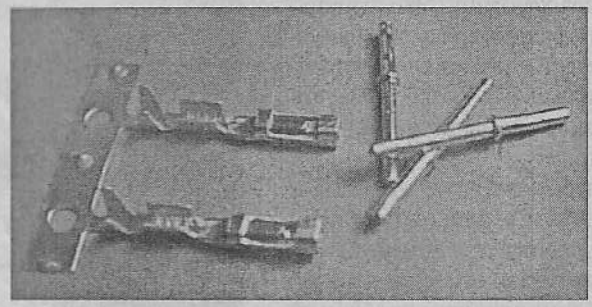


**Рис. 3.9.** Штырьковые выводы для печатных плат создают соединения с помощью нажимных гнезд, поставляемых на несущей ленте (их необходимо отломать)

Нажимные гнезда отламываются от ленты, обжимаются вокруг оголенного конца соединительного провода и припаиваются к нему. Контакт будет хорошим ввиду золотого покрытия и пружинных гнезд.

Конструкции целого ряда контроллерных плат используют именно этот вид соединений. Он позволяет изменять характер подключения к микроконтроллеру периферийных устройств в процессе разработки робототехнических электронных систем.

Цокольные штекеры и гнезда — хороший способ подключения модулей, имеющих два или больше соединений. Штекер припаивается к плате, а гнезда обжимаются и припаиваются к соединительным проводам (рис. 3.10).

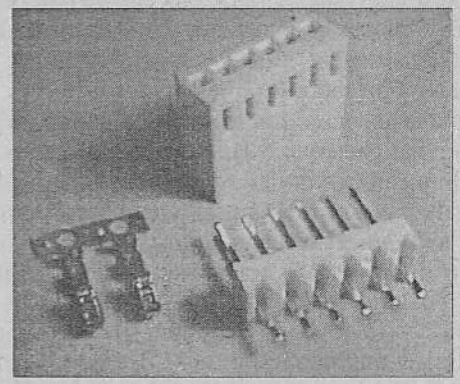


**Рис. 3.10.** Шестиконтактный цоколь (вверху) содержит отдельные гнезда (слева). Он надевается на штекер (справа), который припаивается к плате

Штекеры предлагаются в вариантах с двумя, четырьмя, шестью и восемью контактами. Штекеры и разъемы имеют ориентирующие выступы, в силу чего соединить их неправильно невозможно. Это означает, что очень важно сразу же распаять их корректно, поскольку отдельные гнезда извлекать из корпуса после того, как они были вставлены, очень сложно.

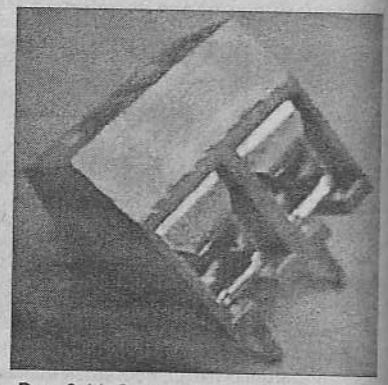


**Рис. 3.11.** Зажимной контакт с двумя клеммами. Существуют также варианты с тремя и четырьмя клеммами. Расстояние между выводами — 5 мм (две полосы)

Зажимные контакты печатной платы (рис. 3.11) — еще один способ создания надежных соединений, которые можно легко отсоединять. На наш взгляд, они удобны для линий питания, поскольку к одной и той же клемме можно подключить несколько проводов, реализовав цепочку подключений линий питания от платы к плате.

## Источники питания

Различные типы источников питания (батареи и блоки питания) уже обсуждались ранее, сейчас же мы поговорим о том, какое напряжение должен обеспечивать источник питания, а также углубимся в технические подробности его реализации.

Первый этап планирования источника питания — составление списка устройств и схем системы, а также их потребностей с точки зрения электропитания. Все системы содержат один или несколько микроконтроллеров PIC, так что имеет смысл начать именно с них. Последние модели PIC работают при любом напряжении питания в диапазоне от 2 до 5,5 В. Такой диапазон дает значительную гибкость, хотя, к сожалению, не охватывает напряжение 6 В, соответствующее четырем щелочным батарейкам. Ближайшее приемлемое напряжение 4,8 В дают четыре NiMH (или NiCd) элемента. На практике, несмотря на номинал одного элемента 1,2 В, в полностью заряженном состоянии он выдает 1,3 В.

Система почти наверняка будет содержать как минимум один двигатель. Двигатель обычно выбирают по его габаритам и скорости вращения (и, возможно, — цене). При условии, что его рабочее напряжение не превышает 12 В, мы или подаем или 12 В, или используем более низкое напряжение, при котором двигатель работает достаточно быстро.



Другие устройства в системе могут предъявлять особые требования к электропитанию. Например, логические КМОП-схемы серии 4000 требуют напряжение в диапазоне 3..15 В, что легко выполнимо, в то время как микросхема 74HC нуждается в напряжении 2..6 В. Она не может работать от того же напряжения питания, что и двигатель, требующий 12 В. Некоторые из полупроводниковых зуммеров и звуковых генераторов имеют широкий диапазон рабочих напряжений, в то время как другие — нет, поэтому всегда проверяйте этот аспект перед покупкой устройства.

Если все компоненты могут работать при одном напряжении, то это намного упростит разработку схемы (рис. 3.12). Вот почему предпочтительнее низковольтные (на 3 В или 6 В) двигатели, которые могут работать при том же самом напряжении питания, что и микроконтроллер PIC. Однако иногда без двигателя на 12 В, соленоида или реле никак не обойтись. В таком случае придется использовать два источника питания (рис. 3.13).

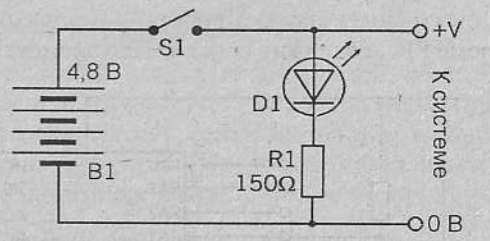



**Рис. 3.12.** Система с одним питающим напряжением. Источник в данном случае — батарея из четырех элементов NiMH. S1 — монтируемый на плате переключатель. D1 — светодиод стандартной яркости. Резистор ограничивает ток, протекающий через светодиод, до уровня примерно 20 мА

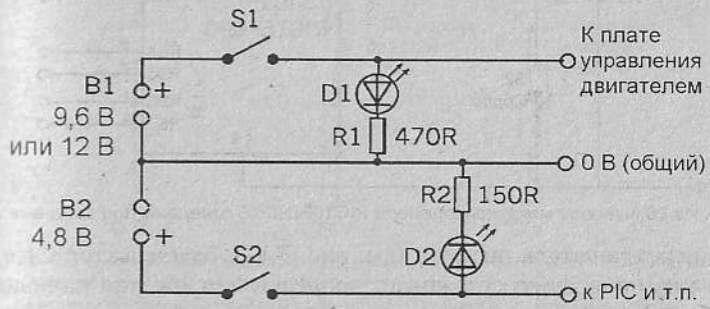



**Рис. 3.13.** Система с двумя питающими напряжениями (портальный робот). Присутствует переключатель для каждого источника питания, а шина 0 В — общая для обоих источников. Обратите внимание на различие сопротивлений R1 и R2

Цепь питания (одинарная или двойная) должна также содержать переключатель и, желательно, — светодиод индикации питания. Некоторые схемы для этого показаны далее.

Если приводные двигатели имеют собственный источник питания, то рекомендуем использовать для него отдельный переключатель. Благодаря этому, можно будет протестировать микроконтроллер PIC, не заставляя робот сновать туда-сюда.

### Расчет сопротивления

Падение прямого напряжения на проводящем светодиоде составляет около 2 В. Падение напряжения на резисторе равно ($V_{пит}$ – 2). Если ток через светодиод должен быть равен *i* ампер, то сопротивление должно быть равно ($V_{пит}$ – 2)/*i*.

Пример: если $V_{пит}$ = 4,8 В, а *i* = 0,02 А, то сопротивление R = (4,8 – 2)/0,02 = = 2,8/0,02 = 140 Ом.

Используйте ближайший номинал 150 Ом.

## Схема включения микроконтроллера

Поскольку электронная система робота — модульная, плата микроконтроллера, кроме PIC, содержит совсем мало элементов (рис. 3.14).

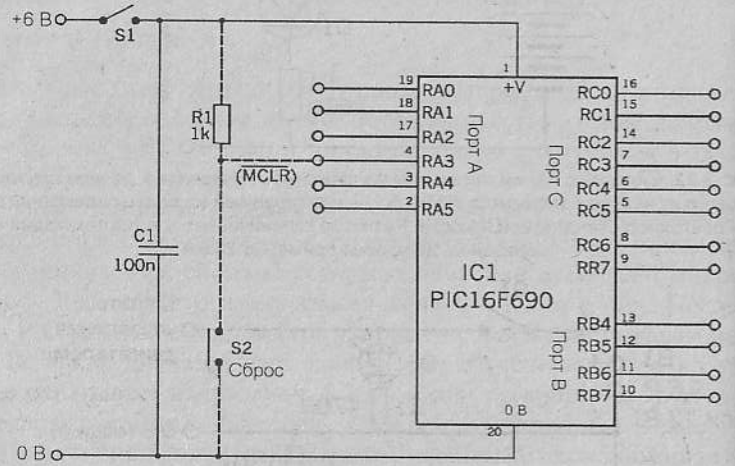



**Рис. 3.14.** Из 20 выводов микроконтроллера PIC16F690 18 отведены под ввод-вывод

S1 — переключатель питания (см. рис. 3.12). Конденсатор C1 с диэлектриком из полиэфира сглаживает всплески напряжения на положительной шине питания.

При первой подаче напряжения каналы RA0..RA2, RA4, RB4, RB5, RC0..RC3, RC6 и RC7 — аналоговые входы. Остальные каналы — цифровые входы. Аналоговые каналы могут быть определены индивидуально как цифровые, а все каналы за исключением RA3 могут быть определены как выходы. Выходы всегда цифровые.



Когда каналы портов A и B (за исключением RA3) сконфигурированы как входы все они могут иметь слабое подтягивающее сопротивление, действующее как высокоомный резистор между входным выводом и линией питания. Вход читается как лог. 1, если только он жестко не заземлен на шину 0 В.

Канал RA3 — исключение. Он может быть сконфигурирован как цифровой вход без слабого подтягивающего сопротивления. Соответствующая цепь показана на рис. 13.14 пунктиром. Она требует внешний резистор. Если канал сконфигурирован как /MCLR для сброса микроконтроллера, то он автоматически получит слабое подтягивающее сопротивление и потребность во внешнем резисторе отпадет.

## Входные цепи

### Одноразрядный вход

На рис. 3.15 показаны основные типы цепей, обеспечивающих цифровые (единичные или нулевые) входы контроллера. Простейший вариант — простой переключатель между выводом и шиной 0 В. Нормальное состояние входа — высокий уровень напряжения, который изменяется на низкий по нажатию кнопки.

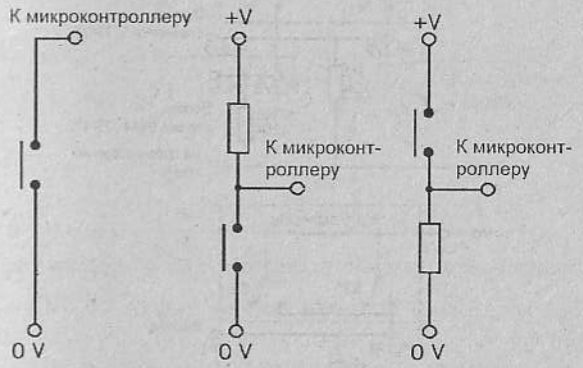



**Рис. 3.15.** Входные цепи для одного разряда. Слева — когда канал имеет слабое подтягивающее сопротивление. На входе — высокий уровень напряжения. Переход в низкий уровень — при нажатии кнопки. В центре — для каналов без слабых подтягивающих сопротивлений. Приемлем резистор на 10 кОм. Нормальное состояние входа — высокий уровень. Справа — то же самое за исключением того, что нормальное состояние входа — низкий у

Для того чтобы использовать эту схему, у канала должно быть активно слабое подтягивающее сопротивление. Это означает, что он должен относиться к порту A или B. Переключатель изображен в виде кнопки, однако использовать можно различные виды переключателей: тумблеры, микропереключатели (часто используемые как концевые), переключатели, срабатывающие по наклону, термопереключатели, герконы, реле и даже прижимные устройства. Все они могут обеспечивать прямой входной сигнал для микроконтроллера.

Иногда входной канал "зашумлен", воспринимая помехи от двигателей и других электромагнитных устройств. Всплески напряжения могут преодолеть слабые подтягивающие сопротивления — особенно, если входная линия длиннее 10 см. В таких случаях безопаснее использовать значительное подтягивающее или согласующее сопротивление в виде резистора на 10 кОм (см. рис. 3.15).

В роботе со множеством подвижных частей может использоваться много микропереключателей, действующих как концевые выключатели. Они необходимы, однако для каждого из них требуется отдельный канал ввода-вывода. Возможно, некоторые из этих каналов потребуются для других целей, и один из способов экономии в данном случае — ввести схему логического "ИЛИ" для некоторых входов (рис. 3.16).

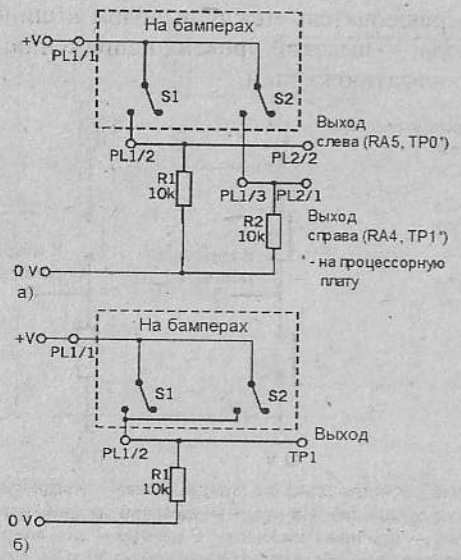



**Рис. 3.16.** *а* — подключение к процессору двух переключателей с отдельными входами. *б* — выходы переключателей объединяются схемой логического "ИЛИ" в один канал



В данном примере робот оснащен двумя бамперами: передним и задним. В обычных условиях они не могут сработать одновременно. С помощью схемы, показанной на рис. 3.16*б*, можно применить логическую операцию "ИЛИ" к выходами переключателей. Когда S1 или S2 замкнут, то канал переходит в единичное состояние. Робот может определить, какой из переключателей сработал, по направлению движения (вперед или назад).

Такая логика предполагает, что нет никакого другого робота, который может врезаться в бампер первого робота. Если же такой робот может присутствовать, то следует возвратиться к схеме на рис. 3.16*а*.

Цифровой входной канал может получать логический входной сигнал от вентиля КМОП или другого КМОП-выхода. Это часто бывает полезным, если выход датчика не переходит полностью из нулевого в единичное состояние. Вентиль формирует сигнал, генерируя четкие логические уровни, понятные микроконтроллеру PIC.

На рис. 3.17 показан фотодиод, подключенный к цифровому входному каналу без слабого подтягивающего сопротивления. Диод — обратносмещенный, так что через резисторы протекает лишь небольшой ток утечки. Переменный резистор используется для регулировки напряжения на резисторах.

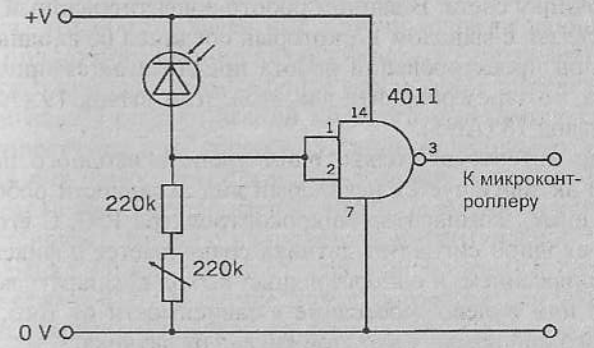



**Рис. 3.17.** Одноразрядный вход от фотодиода (видимого или инфракрасного света)

В случае КМОП-логики любой входной сигнал, уровень которого превышает половину напряжения питания, считается сигналом высокого уровня, а сигнал, уровень которого ниже половины напряжения питания, — сигналом низкого уровня. На рис. 3.17 показан двухвходовой логический элемент "НЕ-И" с объединенными входами. Таким образом, он действует как логический элемент "НЕ" или инвертор. По мере усиления потока света, падающего на фотодиод, возрастает ток утечки, и

напряжение на входах логического элемента увеличивается. Выход изменяется из состояния высокого уровня в состояние низкого уровня на полпути роста напряжения.

Этот же эффект можно было бы получить и без логического элемента с помощью внутреннего компаратора микроконтроллера PIC, однако в некоторых случаях использование внешней логики предпочтительнее.

### Аналоговый входной сигнал

Аналоговый выходной сигнал датчика, наподобие фотодиода (см. выше) или фоторезисторной цепи, можно считать с помощью внутреннего аналого-цифрового преобразователя (АЦП) микроконтроллера PIC. Он выдает 10-разрядный результат, так что данные с датчика можно получать с высокой точностью. Таким образом, микроконтроллер PIC можно запрограммировать на обработку различных уровней освещенности.

Пример использования АЦП — робот "Скутер" (см. главу 6), который непрерывно контролирует уровень освещенности перед собой, пока не определит направление на самый яркий источник света в поле его зрения. После этого он начинает двигаться вперед по направлению к этому источнику света. В данном роботе ориентированный вперед фоторезистор связан с выводом 19, который соединен со входным каналом AN0. Если при проектировании робота предполагается прием аналогового сигнала, то зарезервируйте для этой цели вывод 19 (AN0) и, возможно, — вывод 18 (AN1).

Если нас интересует только один уровень входного напряжения, при котором активизируется некоторый вид активности робота, то следует использовать компаратор микроконтроллера PIC. С его помощью аналоговый входной сигнал от датчика сравнивается с фиксированным опорным напряжением, и одноразрядный выход компаратора переходит в единичное или нулевое состояние в зависимости от того, выше или ниже опорного напряжения входной сигнал от датчика.

Опорное напряжение можно формировать внутренне и программировать его, или же оно может быть сгенерировано внешне. Внешнее опорное напряжение может поступать от специального устройства или от цепи делителя (рис. 3.18). Первый из названных вариантов обеспечивает фиксированное напряжение с высокой точностью, а второй дает фиксированную *долю* напряжения питания. Делитель напряжения зачастую предпочтительнее, поскольку его можно регулировать с помощью простой отвертки.



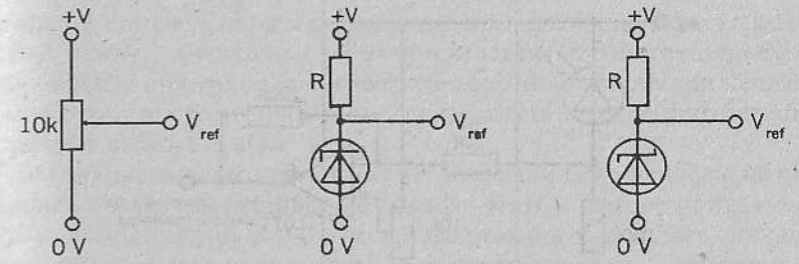



**Рис. 3.18.** Источники опорного напряжения. Слева — переменный выход потенциометра пропорционален напряжению питания. В центре — стабилитрон обеспечивает стабильное напряжение при изменении напряжения питания. Справа — схема формирования опорного напряжения на основе ширины запрещенной зоны (обеспечивает более высокую точность)

**Расчет R**

1) $I_{max}$ = требуемый ток (в мА) + 5.
2) Минимальная номинальная мощность = $V_{ref}$ x $I_{max}$.
3) Резистор R = $(V_{пит} - V_{ref})/I_{max}$.

В случае с АЦП входное напряжение, когда оно равно опорному, дает на выходе значение 3FFh (если считываются все десять разрядов ADRES и ADRESH). Если в качестве опорного используется напряжение питания, это обеспечивает высокоточную разрешающую способность даже при преобразовании низких входных напряжений.

В случае с компаратором опорное напряжение определяет уровень входного сигнала, при достижении которого изменяется состояние на выходе. Существует два способа установки опорного напряжения, и первый из них заключается в использовании разрядов 0..3 регистра VRCON. Этот метод обеспечивает только 16 возможных установок. При попытке установить опорное напряжение на критический уровень, некоторый перепад может оказаться слишком низким, а следующий — слишком высоким. В этом случае может потребоваться усиление входного сигнала перед его подачей на компаратор. Схема на рис. 3.19 иллюстрирует способ повышения различительной способности датчика.

Датчик в данном случае — это обратносмещенный фотодиод, последовательно с которым включено переменное сопротивление для настройки на работу в условиях высокой или низкой освещенности. По существу схема с операционным усилителем (ОУ) — это инвертирующий усилитель. Его коэффициент усиления определяется двумя фиксированными резисторами и равен 100 000 / 10 000 = 10. Небольшое изменение напряжения от датчика теперь велико по сравнению с перепадами внутреннего источника опорного напряжения.

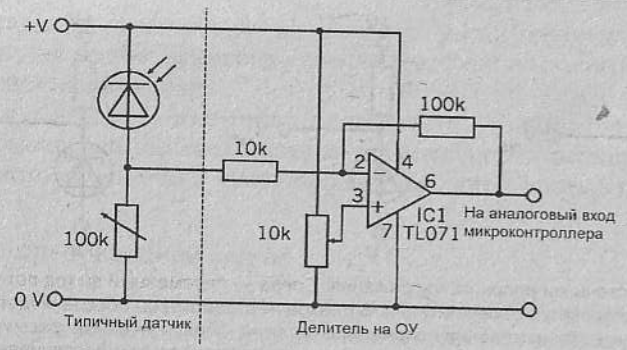



**Рис. 3.19.** Схема масштабирования на операционном усилителе обрабатывает выходное напряжение датчика перед передачей его в микроконтроллер PIC

Переменный резистор на 10 кОм устанавливает уровень нуля. Обратите внимание, что выход схемы — инверсный.

### Датчики

Робот должен знать, что происходит в окружающем мире, поэтому все наши роботы оборудованы несколькими датчиками, связанными с микроконтроллером. Рассмотрим датчики, часто используемые в робототехнике.

**Резистивные датчики** реагируют на изменения в некоторой количественной характеристике, наподобие освещенности или позиции, путем изменения сопротивления. Такое изменение легко измерить, пропустив ток через датчик, что сформирует изменяющееся напряжение, которое передается микроконтроллеру. Обычно в качестве схемы опроса сигнала выступает делитель напряжения с датчиком в качестве одного из резисторов.

**ЭДС-датчики** реагируют на изменения в контролируемой характеристике путем изменения ЭДС (электродвижущей силы, или, грубо говоря, напряжения), которую они производят. Полученный сигнал передается в микроконтроллер.

При работе с датчиками доступно широкое поле для фантазии. Так, например, робот "Скутер" излучает перед собой луч света, а затем обнаруживает свет, отраженный от объектов, перекрывающих его путь. В этом случае фотоэлемент используется как датчик приближения.

### Фотоэлементы

Наиболее популярный фотоэлемент в наших роботах — это **фоторезистор**, который, как следует из его названия, представляет собой ре-



зистивный датчик. Сопротивление типичного фоторезистора, наподобие GRP12, лежит в диапазоне от 1 мОм или выше в темноте до приблизительно 80 Ом при ярком солнечном свете. В помещении при непрямом дневном или искусственном освещении сопротивление фоторезистора составляет несколько кОм.

Фоторезисторы реагируют на свет большинства оттенков с пиковой реакцией на желтый цвет. Из всех фотоэлементов фоторезисторы — самый медленнодействующие. Их время реакции составляет несколько десятков или сотен миллисекунд. Хотя для людей это — достаточно быстро, микроконтроллер PIC работает гораздо быстрее. В программах могут потребоваться короткие задержки, чтобы фоторезистор адаптировался к освещенности.

Делитель напряжения (рис. 3.20) может содержать постоянный резистор, переменный резистор или и то, и другое. Переменный резистор позволяет устанавливать выходное напряжение для любого уровня освещенности. Общее сопротивление должно быть в том же диапазоне, что и среднее сопротивление фоторезистора в предполагаемых условиях работы.

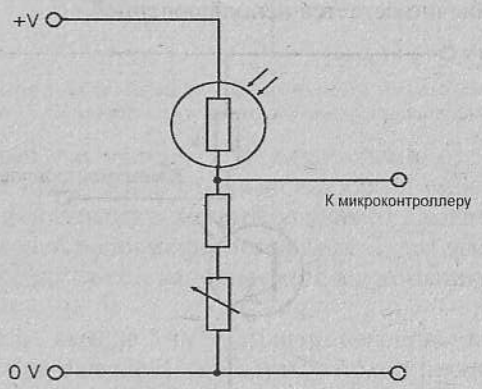



**Рис. 3.20.** По мере увеличения освещенности уменьшается сопротивление фоторезистора и увеличивается напряжение, передаваемое в микроконтроллер

Другой популярный фотоэлемент — **фотодиод**. Его действие основано на том факте, что ток утечки, когда диод смещен в обратном направлении, изменяется в зависимости от интенсивности освещенности (см. рис. 3.19). Ток утечки очень невелик. В темноте он составляет всего лишь несколько наноампер и возрастает приблизительно до 1 мА в условиях яркой освещенности. Сопротивление резистора составляет несколько сотен тысяч ом, поэтому ток утечки формирует на нем прием-

лемое напряжение. Обычно подходящее выходное напряжение обеспечивает резистор на 330 кОм. Во избежание понижения напряжения выход должен быть связан с высокоомным входом. Микроконтроллеры PIC — это КМОП-устройства, и потому их входы — высокоомные.

Фотодиоды в общем случае лучше реагируют на свет в красном конце спектра. Некоторые из них особенно чувствительны к инфракрасному излучению. Такие фотодиоды используют совместно с инфракрасными светодиодами для считывания информации с оптических кодеров. Они также задействованы в роботах в качестве датчиков отслеживания линии, поскольку менее подвержены влиянию внешних источников видимого света.

Время реакции фотодиода очень незначительно (в общем случае — несколько сотен наносекунд), поэтому никаких проблем с ним не возникает.

**Фототранзистор** (рис. 3.21) по своим свойствам подобен фотодиоду, хотя характеризуется более длительным временем реакции. Он подключается так же, как n-p-n-транзистор в усилителе с общим эмиттером. В фототранзисторах часто отсутствует вывод базы, а если он и присутствует, то база обычно остается неподключенной.

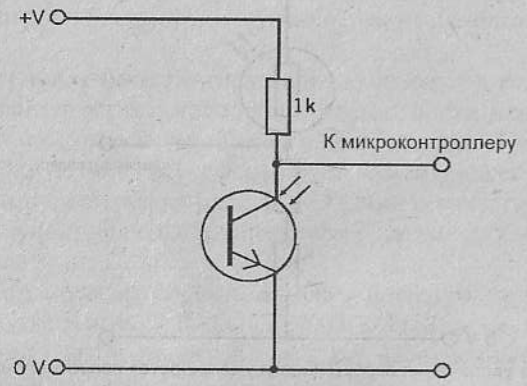



**Рис. 3.21.** У некоторых фототранзисторов отсутствует вывод базы. Это — двухвыводные устройства с коллектором и эмиттером

Фототранзистор может управлять током базы благодаря тому, что падающий на него свет высвобождает подачу электронов, действующих как ток базы.

Для повышения чувствительности фототранзисторы часто совмещают на одном кристалле с усилительной схемой или парой Дарлингтона. Существуют аналогичные устройства и на основе фотодиодов.



### Цифровые выходные данные

Аналоговый сигнал от фотоэлемента обычно обрабатывается встроенным компаратором микроконтроллера PIC, однако иногда этого недостаточно. В любом случае проще обработать момент срабатывания по перепаду уровня аппаратно. Соответствующая схема использует компаратор на ОУ для преобразования аналогового выходного сигнала в цифровой (рис. 3.22).

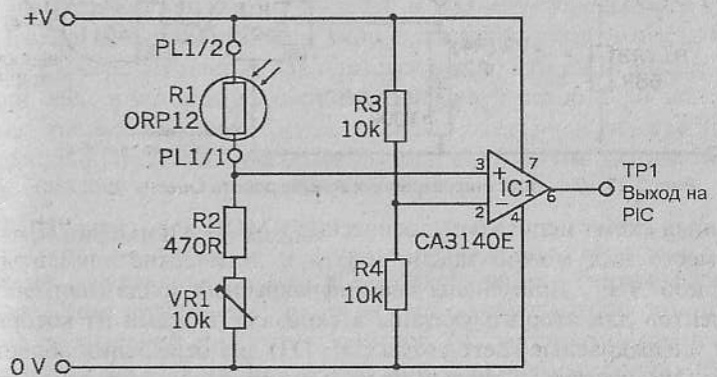



**Рис. 3.22.** На выходе этой схемы формируется высокий уровень сигнала, когда сила освещенности превосходит некоторый предустановленный порог

Операционный усилитель — с двумя входами, без обратной связи, поэтому разница между входными напряжениями умножается на коэффициент усиления открытого контура усилителя. Когда напряжение на выводе 3 превышает напряжение на выводе 2, на выходе получается очень резкий перепад от состояния низкого к состоянию высокого уровня.

Напряжение на выводе 3 всегда равно половине напряжения питания. Напряжение на выводе 2 варьируется прямо пропорционально освещенности. Оно устанавливается путем настройки VR1 на половину напряжения питания, когда фоторезистор освещается светом, соответствующим уровню срабатывания.

Эта схема используется для фоторезисторного фотоэлемента в роботе "Искатель" (см. главу 9). Размах сигнала на выходе CA3148E близок к значению напряжения питания, что обеспечивает четкий сигнал для микроконтроллера.

Робот "Искатель" демонстрирует еще один способ формирования цифровых выходных данных. Логические элементы КМОП изменяют свое состояние, когда уровень входного напряжения близок к половине

напряжения питания. Они действуют подобно компаратору с опорным напряжением $V_{пит}/2$ (рис. 3.23).

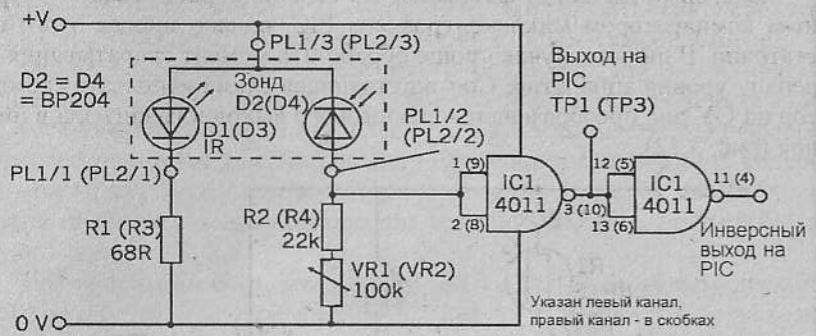



**Рис. 3.23.** Цепь двух инфракрасных зондов робота Quester (стр. 258).

Данная схема использует логические КМОП-элементы "НЕ-И", однако вместо них можно задействовать и логические элементы "НЕ-ИЛИ" либо "НЕ". Применены два инфракрасных зонда (наименования компонентов для второго указаны в скобках), каждый из которых содержит инфракрасные светодиоды (D1, D3) для освещения области под зондом. Отраженные инфракрасные лучи распознаются парой инфракрасных диодов BP204. Обратите внимание, что полярности двух диодов противоположны.

### Распознавание цвета

Роботу может потребоваться определить цвет объекта. Например, он может решать задачу сортировки разноцветных строительных блоков. Простой способ реализовать это — поочередно освещать объект лучами двух или трех различных оттенков. Количество света, отраженного от объекта, измеряется одним фоторезисторным фотоэлементом.

Для проверки этой методики маленький (30 мм) квадрат цветной бумаги освещался красным, зеленым и синим светодиодом высокой интенсивности. Фоторезистор, подвергавшийся воздействию света, отраженного от этих квадратов, находился в составе схемы, показанной на рис. 3.20, а его выходное напряжение подавалось на компаратор. После соответствующей установки опорного напряжения на выходе компаратора был считан сигнал для каждого цвета бумаги и каждого цвета светодиода. Низкой интенсивности отраженного света соответствует лог. 0, а высокой — лог. 1. Типичные результаты сведены в табл. 3.2.



**Таблица 3.2.** Эксперимент по реакции на отраженный свет

| Цвет бумаги | Белый | Красный | Зеленый | Голубой | Черный |
|---|---|---|---|---|---|
| Красный светодиод | 1 | 1 | 1 | 0 | 1 |
| Зеленый светодиод | 1 | 0 | 1 | 0 | 0 |
| Голубой светодиод | 1 | 0 | 0 | 1 | 0 |

Как видим, совпадает только реакция датчика на красную и черную бумагу. Если микроконтроллер PIC запрограммировать таким образом, чтобы светодиоды мигали по очереди, и при этом каждый раз опрашивался выход компаратора, то можно идентифицировать цвета бумаги. Проблема с черным цветом заключается в том, что он отражает инфракрасный свет, к которому чувствителен фоторезистор. При зеленом освещении красная бумага отражает меньше света, чем черная, поэтому при правильно подобранном опорном напряжении эти цвета можно различить.

### Фотоприемник с памятью

Схема, показанная на рис. 3.24, обнаруживает быстрые световые вспышки. Хотя подобные события можно обрабатывать с помощью механизма прерываний микроконтроллеров PIC, вмешательства в ход работы программы в непредсказуемые моменты времени лучше избегать. Данная схема позволяет микроконтроллеру опрашивать выход для обнаружения вспышки света.

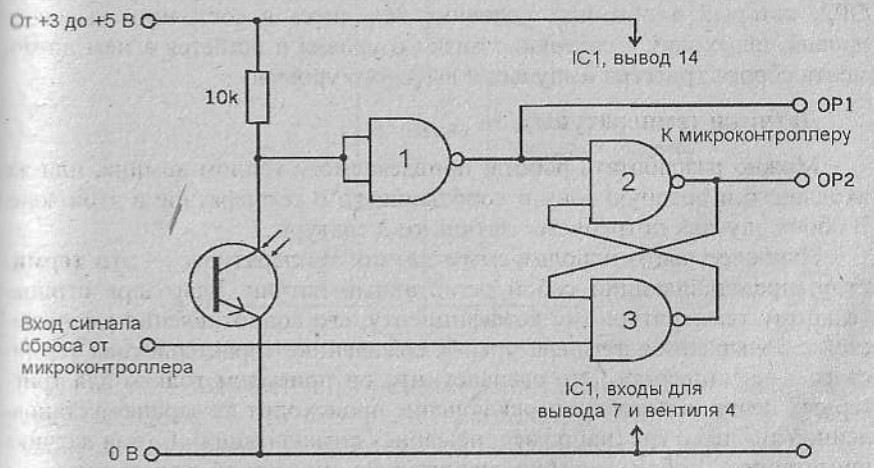



**Рис. 3.24.** Датчик световых вспышек на фототранзисторе и RS-триггере

Фототранзистор — предпочтительно на паре Дарлингтона. При вспышке он на короткое время переходит в проводящее состояние, и напряжение на его коллекторе падает. Этот импульс низкого уровня инвертируется вентилем 1 и подается через вывод OP1 на микроконтроллер. Если входной канал сконфигурирован на прерывание по изменению состояния, то генерируется прерывание.

Вентили 2 и 3 формируют бистабильный RS-триггер, состояние которого изменяется по импульсу низкого уровня на входе установки (S) или сброса (R). Эта часть схемы предназначена для обнаружения разрывов в потоке обычного света, падающего на фототранзистор. Такой разрыв формирует импульс высокого уровня, который инвертируется вентилем 1 и переключает триггер из состояния сброса (низкий уровень на выводе OP2) в состояние установки (высокий уровень на выводе OP2). Однажды перейдя в состояние высокого уровня, вывод будет оставаться в нем до сброса триггера. В это время микроконтроллер периодически опрашивает вывод OP2, а затем подает на вход сброса, который в обычном случае находится в состоянии высокого уровня, короткий импульс низкого уровня. В результате на выводе OP2 устанавливается сигнал низкого уровня.

Реакция схемы будет противоположной, если вентили "НЕ-И" заменить логическими элементами "НЕ-ИЛИ" (микросхема 4001 или 74HC02). В результате при вспышке света вывод OP1 переходит в состояние высокого уровня. Кроме того, срабатывает триггер, и вывод OP2, который в обычных условиях находится в состоянии высокого уровня, переходит в состояние низкого уровня и остается в нем до момента сброса триггера импульсом высокого уровня.

### Датчики температуры

Можно разработать робота, привлекаемого теплом камина, или же входящего в опасную зону и сообщающего о температуре в этой зоне. В обоих случаях потребуется датчик температуры.

Наиболее часто используемым датчик температуры — это **термистор**, представляющий собой резистивный датчик. Благодаря отрицательному температурному коэффициенту, его сопротивление уменьшается с повышением температуры. К сожалению, характеристика термистора — нелинейная. Это означает, что он приемлем только для триггерных цепей, в которых переключение происходит по заранее установленной температуре (например, пожарная сигнализация). Схема датчика представляет собой делитель напряжения, подобный показанному на рис. 3.20, но с заменой фоторезистора термистором.



Термисторы доступны в виде стержней, дисков и шариков из резистивного материала. Их сопротивление обычно указывается для комнатной температуры (25°C). Термисторы изготовляют в большом диапазоне номиналов от 25 Ом до 1 МОм при допуске 10% или 20%. Поскольку их точность невелика, триггерная цепь нуждается в подстроечном резисторе, включенном последовательно с термистором для коррекции температурной установки.

#### *Формула для термистора*

Зависимость сопротивления R термистора от температуры T задается формулой:

$$R_T = R_{ref} e^{-\beta(1/T - 1/Tref)}$$

где $R_{ref}$ — сопротивление при температуре $T_{ref}$ (температура в градусах Кельвина); e — экспоненциальная константа 2,718, а β — коэффициент, зависящий от структуры термистора. В типичном случае он равен 4 000, однако его фактическое значение указывается в техническом описании термистора.

Устройство для точного измерения температуры — **зонный датчик**, который имеет три вывода и выглядит как транзистор. В качестве примера можно привести датчик LM35CZ, который работает при любом напряжении питания в диапазоне 4..20 В (рис. 3.25).

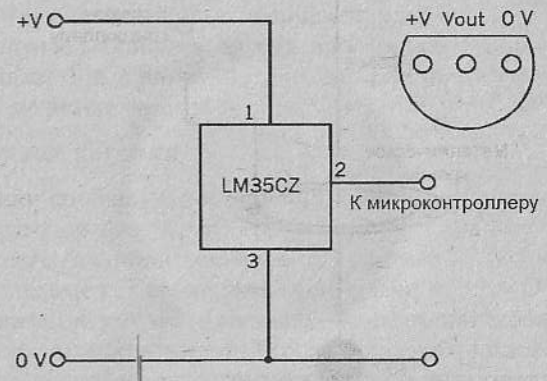



**Рис. 3.25.** Температурный зонный датчик не требует внешних компонентов

На выходе получается напряжение, пропорциональное температуре в градусах Цельсия. По мере роста температуры от 0°C до 110 °C выходной сигнал изменяется от 0 В до 1,1 В. Что может быть проще?

Точность LM35CZ составляет ±0,4°C при 25°C и не превышает ±0.8°C во всем диапазоне измерений.

### Датчики движения

**Переключатель по наклону** позволяет предотвратить катастрофу робота на неровной или наклонной поверхности. Он смонтирован на роботе таким образом, чтобы в нормальном состоянии находиться в вертикальном положении. В таком положении переключатель разомкнут, и замыкается при наклоне корпуса робота всего лишь на несколько градусов.

Самый простой способ подключения переключателя по наклону — цифровой канал со слабым подтягивающим сопротивлением. При срабатывании переключателя вход переходит в состояние низкого уровня.

"Родственники" переключателей по наклону — **виброперекючатели**, замыкающие и размыкающие контакт при малейших возмущениях. Они предназначены для использования в системах безопасности, однако применяются также и в робототехнике.

Собрать переключатель по наклону в домашних условиях очень просто (рис. 3.26). При этом он может работать даже лучше, чем заводское изделие.

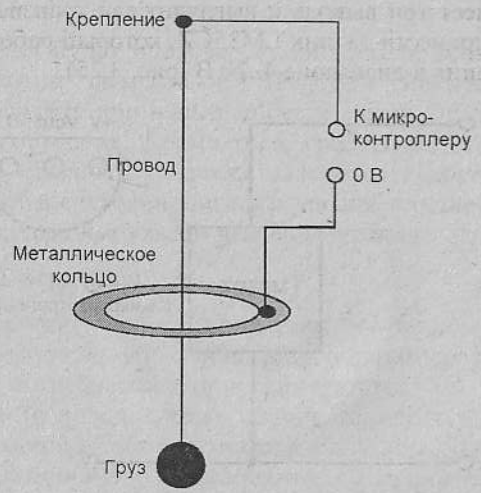



**Рис. 3.26.** Когда переключатель наклоняется, провод маятника касается металлического кольца. Это замыкает цепь, и на входной канал микроконтроллера подается 0 В

Основная проблема заключается в том, что маятник может продолжать раскачиваться даже тогда, когда переключатель больше не наклонен. Его движение должно необходимо погасить. Попытайтесь найти провод из пружинного металла или используйте проволочную пружину.



Пружина растяжения (в ней смежные витки касаются друг друга) предпочтительнее пружины сжатия.

Другая проблема с переключателями по наклону заключается в том, что они могут срабатывать, когда робот ускоряет или замедляет движение. Решение — реализовать в программе кратковременное игнорирование сигнала от переключателя сразу же после включения или выключения приводных двигателей.

Переключатель по наклону просто выдает цифровой сигнал, когда угол наклона робота превышает некоторое фиксированное значение, но **датчик наклона** обеспечивает аналоговую величину, который представляет собой *меру* угла (рис. 3.27).

Жесткий, ориентированный вниз, рычаг закреплен на валу переменного потенциометра. Масса на его нижнем конце достаточна для поворота вала при наклоне робота. Напряжение на подвижном контакте потенциометра изменяется в зависимости от угла и может считано АЦП микроконтроллера PIC.

Это устройство чувствительно к наклону в одной вертикальной плоскости, перпендикулярной валу. В случае наклона в других плоскостях оно может выдавать неправильные показания или вообще никак не реагировать.

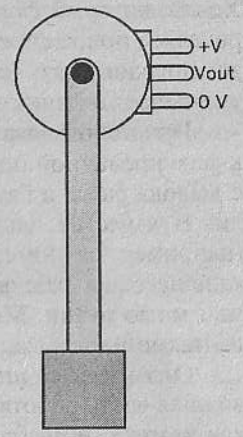



**Рис. 3.27.** Выходное напряжение датчика наклона изменяется в зависимости от угла

### Локационные датчики

Эти датчики сообщают роботу о положении в пространстве его самого или некоторой его части. Например, портальному роботу очень важно точно знать позицию инструмента на раме X. Без этой информации он просто не сможет выполнять свои задачи.

Для локализации робота (или чаще — некоторой его части) в диапазоне нескольких десятков миллиметров можно использовать методику, основанную на использовании **линейного потенциометра** — переменного резистора ползункового типа, наподобие тех, которые часто используются для регулировки частотной характеристики звукового усилителя. Концы его проводящей дорожки подключены к положительному полюсу источника питания и к 0 В. Объект прикреплен к ползунку механически. Выходное напряжение на ползунке изменяется пропор-

ционально положению контакта на дорожке. Другими словами, это — прямолинейный сервомеханизм.

Длина проводящей дорожки потенциометра ограничена, из-за чего данный тип датчиков применим только на небольших дистанциях. Для более длинных расстояний можно использовать **маркеры**. Например, маленькие магниты располагаются снаружи вдоль рельса, по которому перемещается рама портального робота. Рама несет датчик на эффекте Холла, который формирует импульс, проходя мимо магнита. Робот определяет положение рамы путем простого подсчета импульсов. Он также должно знать направление движения, и потому увеличивает или уменьшает счетчик импульсов.

При использовании метода маркеров начало движения должно быть в фиксированной позиции. Именно поэтому портальный робот начинает с вывода рамы в базовое положение, используя концевые переключатели. В качестве альтернативы можно применить оптические маркеры (например, равномерно распределенные черные метки). Подвижная деталь несет на себе фотоэлемент, формирующий импульс при прохождении мимо метки. Метки подсчитываются так же, как и в случае с магнитными маркерами.

**Оптические шифраторы** действуют по другому принципу. Подвижная часть робота соединена с прозрачной полосой, на которую нанесен некоторый шаблон. Шаблон состоит из четырех или большего количества строк, содержащих прозрачные и непрозрачные прямоугольники. За полосой расположен массив источников света, а соответствующий массив фотоэлементов обнаруживает свет, проходящий через прозрачные секции (рис. 3.28).

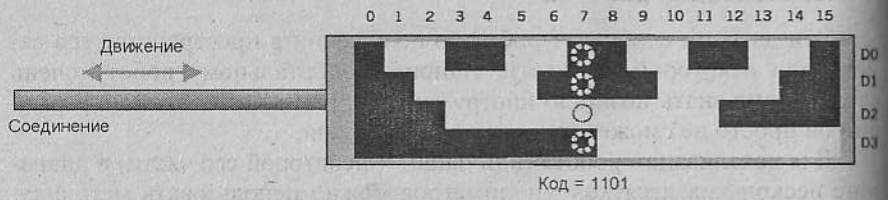



**Рис. 3.28.** Оптический шифратор на шестнадцать позиций. В данном случае активна позиция 7, а в микроконтроллер передается четырехразрядный код 1101

При четырех строках прямоугольник и четырех считывающих датчиках можно получить 16 уникальных состояний. Структура полос основана на коде Грея, в котором четыре бита не следуют нормальной двоичной последовательности счета от 0000 до 1111. В двоичной последовательности две или более цифр могут изменяться одновременно (например с 0111 на 1000). Вероятность того, что все они изменятся *в точности* одновременно (причина ошибок) крайне мала. В коде Грея одновременно изменяется только одна цифра при каждом перемещении от позиции к позиции.



Для определения угловой позиции можно также воспользоваться штрих-кодом в виде концентрических кругов. Эта методика применима, например, для считывания углов между сегментами манипулятора робота. В мобильном роботе она может использоваться для определения угла поворота колес. Зная радиус колес, и предполагая, что они не скользят, мы получаем точное расстояние, пройденное роботом.

Тем не менее, наверное, наиболее популярными локационными датчиками являются концевые переключатели. Они надежны и не создают трудностей при съеме сигнала. Именно поэтому в этой книге описано так много примеров их использования.

### Датчики приближения

Роботу зачастую необходимо знать о приближении к какому-либо препятствию. Один из подходов к получению такой информации заключается в установке бамперов с микропереключателями, фиксирующими физический контакт. Например, такие бамперы использованы в роботе "Искатель".

В общем случае предпочтительнее обнаруживать объекты на некотором расстоянии до них, до реального столкновения. Фотоэлементы робота "Скутер" реализуют это, реагируя на отраженный от преграды свет. Чем интенсивнее отраженный свет, тем ближе объект. Количество отраженного света может также зависеть от размеров и цвета объекта, поэтому данный метод в некоторых обстоятельствах может давать неверные результаты.

Еще одна методика связана с использованием ультразвука, о чем речь пойдет чуть позже.

### Акустические датчики

Схема, показанная на рис. 3.29, оснащена микрофоном для обнаружения звука, усилителем для увеличения чувствительности и триггерной цепью для выдачи на микроконтроллер сигнала высокого уровня при обнаружении звука.

Всплески напряжения на R1 подаются на операционный усилитель, работающий без обратной связи, как компаратор, поэтому его выходное напряжение лежит в широком диапазоне. Усиленный всплеск проходит через C1 на вход триггера, построенного на вентилях "НЕ-И", ввиду че-

го он переключается по спаду напряжения до уровня ниже $V_{пит}/2$. Его выход переключается из состояния низкого уровня в состояние высокого уровня.

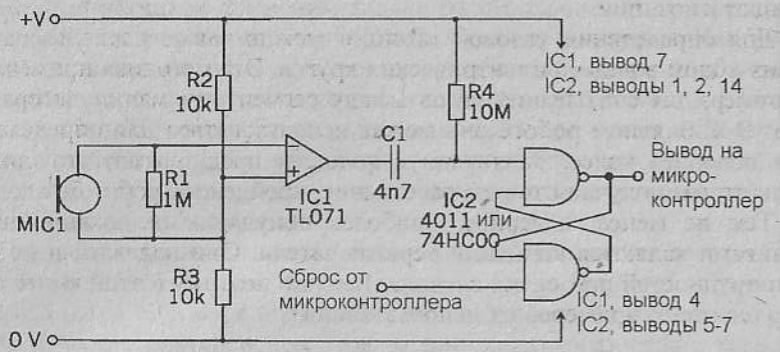



**Рис. 3.29.** Пьезоэлектрический микрофон генерирует всплески напряжения, которые усиливаются и используются для срабатывания триггера

Будучи однажды установлен даже кратковременным понижением напряжения, выход триггера остается в состоянии высокого уровня до тех пор, пока на входе сброса (в обычных условиях находится в состоянии высокого уровня) не появится кратковременный сигнал низкого уровня. Это сбросит триггер, и на его выходе появится сигнал низкого уровня.

Следующий усилитель (рис. 3.30) более сложен, однако гораздо более чувствительный. Он основан на использовании электретного микрофона.

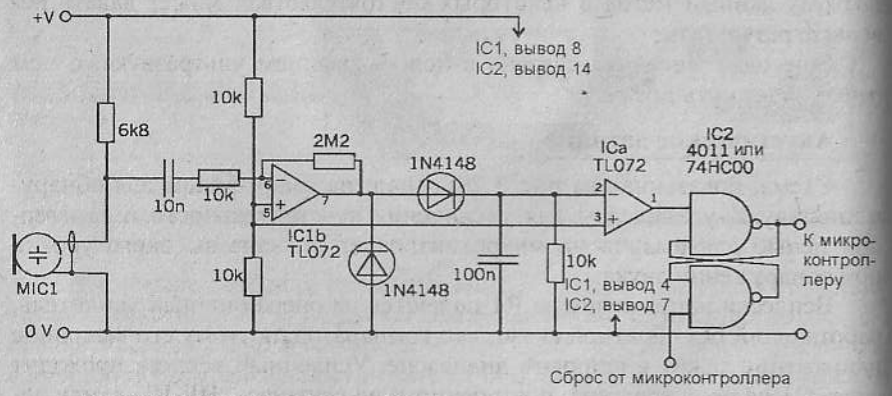



**Рис. 3.30.** Высокочувствительный акустический датчик с цифровым выходом



Звуковой сигнал, сформированный микрофоном, проходит конденсатор и усиливается в 220 000 раз операционным усилителем. Затем он выпрямляется двумя диодами, а на конденсаторе емкостью 100 нФ нарастает положительное напряжение. Оно сравнивается с половинным значением напряжения питания и в случае его превышения второй операционный усилитель, работающий как компаратор, переключает свой выход в состояние низкого уровня, чем вызывает срабатывание триггера.

Сформированное на конденсаторе напряжение постоянно отводится на шину 0 В через резистор, устраняющий влияние очень слабых фоновых шумов. После считывания выхода микроконтроллер сбрасывает триггер обычным способом.

### Ультразвуковые датчики

Ультразвук, под которым в общем случае понимают звук с частотой 40 кГц, имеет два применения в робототехнике: распознавание близости к объекту и измерение расстояния. При этом используются две схемы: генератора и приемника.

Схема генератора (рис. 3.31) содержит осциллятор, построенный на двух логических элементах "НЕ-И". Выход каждого из этих вентилей связан с еще двумя элементами "НЕ-И", входы которых объединены для имитации логического элемента "НЕ". Сигнал на выходе одного из вентилей не совпадает по фазе ровно на 180° с выходным сигналом другого вентиля. Эти два сигнала подаются на кристалл ультразвукового передатчика. Благодаря несовпадению по фазе, они формируют двухтактный цикл, создающий четкий ультразвуковой сигнал.

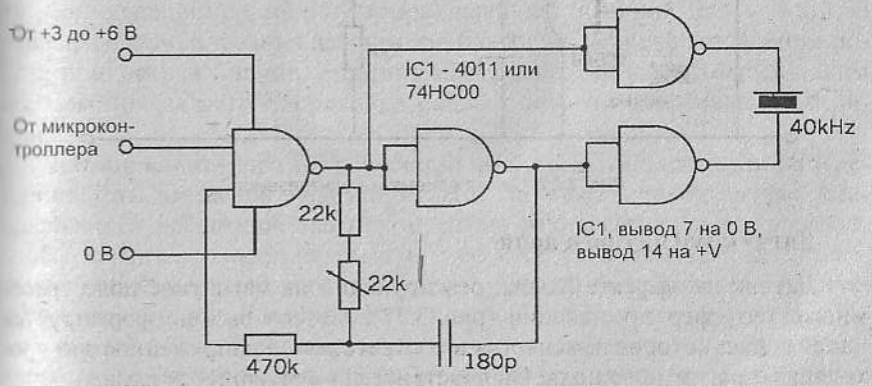



**Рис. 3.31.** Схема генератора ультразвука, построенная на одной логической микросхеме

У генератора присутствует управляющий вход от микроконтроллера PIC, который для активизации генератора переходит в состояние высокого уровня. Это позволяет микроконтроллеру формировать короткие импульсы ультразвука, которые отражаются обратно от объектов на расстоянии одного-двух метров. Время, проходящее между излучением импульса и поступлением в приемник отраженного сигнала, измеряется микроконтроллером. При условии, что скорость звука немного превышает 330 м/с (в зависимости от температуры воздуха), полученное время используется для вычисления расстояния до объекта.

Если генератор используется только для распознавания приближения, а выходных каналов недостаточно, то генератор может работать непрерывно без подключения микроконтроллера на вход вентиля. Вместо этого с логическим элементом соединяются оба входа.

Схема приемника (рис. 3.32) сложнее, поскольку содержит усилитель на двух транзисторах для усиления сигнала от кристалла приемника. Сигнал затем выпрямляется диодом для получения уровня выходного напряжения, падающего при получении ультразвука. Этот аналоговый сигнал подается на компаратор или АЦП микроконтроллера PIC.

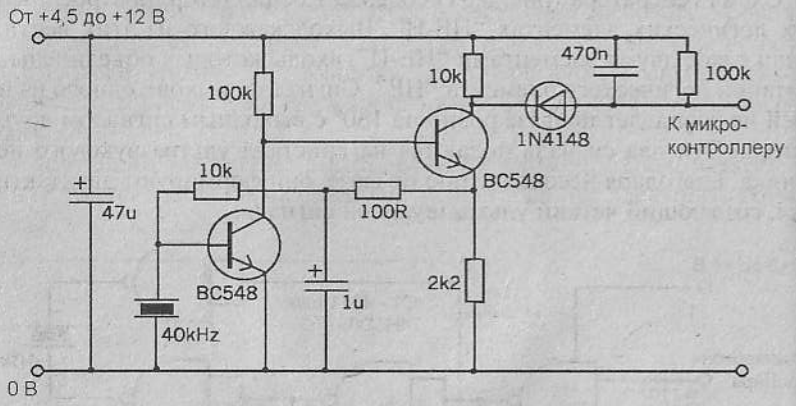



**Рис. 3.32.** Схема приемника ультразвука

### Датчик магнитного поля

Датчик на эффекте Холла, реагирующий на магнитное поле, имеет множество сфер применения (рис. 3.33). На его выходе формируется напряжение, которое изменяется в соответствии с напряженностью проходящего магнитного поля. Он чувствителен к полярности поля.



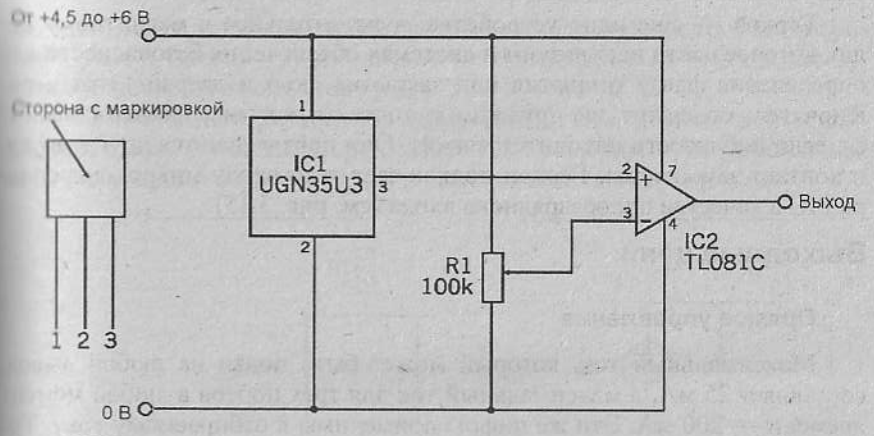



**Рис. 3.33.** Выходной сигнал датчика на эффекте Холла можно измерять как переменное напряжение или преобразовать в цифровой сигнал, как в показанной схеме. Вместо ОУ можно использовать встроенный компаратор микроконтроллера PIC

В качестве датчика приближения он обнаруживает поле небольшого постоянного магнита. Например, портальный робот использует маленькие ферритовые магниты в качестве маркеров, размещенных вдоль рельс. Каждый раз, когда датчик проходит близко к магниту, он посылает сигнал микроконтроллеру. Выходной сигнал, выводимый на микроконтроллер, поднимается или падает в зависимости от полярности магнита.

Данное устройство может также выполнять роль концевого переключателя, обнаруживая момент достижения некоторой подвижной части робота (например, манипулятора) заданной позиции. Это — один из вариантов использования датчика приближения. К подвижной части робота прикреплен магнит, а датчик на эффекте Холла размещен таким образом, чтобы магнит проходил рядом с ним в отслеживаемой позиции.

Датчик магнитного поля также применим в тахометре для измерения частоты вращения вала или колеса. На ободе в таком случае устанавливается небольшой магнит, а датчик размещается таким образом, чтобы магнит проходил рядом с ним по мере вращения колеса. Для каждого оборота генерируется импульс, а частота импульсов (их количество в течение установленного периода времени) используется для измерения скорости вращения. Если колесо — это часть приводного механизма, то микроконтроллер может вычислить скорость и расстояние перемещения робота.

Геркон — еще одно устройство, чувствительное к магнитному полю, которое часто используют в системах обеспечения безопасности для определения факта открытия или закрытия окон и дверей. Этот переключатель содержит два пружинных контакта, которые намагничиваются, если поблизости находится магнит. Они притягиваются друг к другу, и контакт замыкается. Геркон подключается ко входу микроконтроллера PIC в качестве одноразрядного входа (см. рис. 3.15).

## Выходные цепи

### Прямое управление

Максимальный ток, который может быть подан на любой вывод, составляет 25 мА, а максимальный ток для трех портов в любой момент времени — 200 мА. Эти же цифры применимы к отбираемому току. Тока в 25 мА достаточно для управления обычным светодиодом, однако для светодиодов высокой или сверхвысокой яркости необходим ток от 30 до 50 мА или выше.

Примеры цепей прямого управления светодиодом и пьезоэлектрическим громкоговорителем показаны на рис. 3.34 и рис. 3.35.

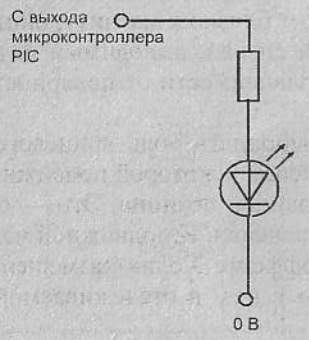



**Рис. 3.34.** Прямое управление светодиодом. Для расчета номинала резистора используйте формулу, рассмотренную выше в разделе "Источники питания"


**Рис. 3.35.** Прямое управление пьезоэлектрическим громкоговорителем

### Транзисторные ключи

Устройства, требующие более 25 мА, управляются транзисторными ключами. При этом чаще всего используют биполярные плоскостные n-p-n-транзисторы, однако n-канальные полевые МОП-транзисторы также применимы. Транзисторные ключи служат для подключения мощных



светодиодов, сирен и зуммеров, двигателей, соленоидов, громкоговорителей, реле и многих других устройств. Далее такие устройства мы будем обобщенно называть "нагрузкой".

Смысл транзисторного ключа (рис. 3.36) в том, что малый ток, втекающий в базу (b) транзистора создает ток примерно в 100 раз большей силы, втекающий в коллектор (c) и вытекающий из эмиттера (e).

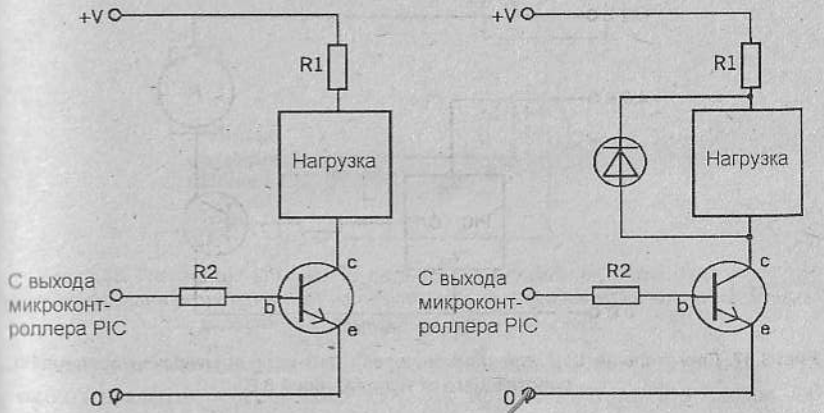



**Рис. 3.36.** Ключи на биполярных плоскостных транзисторах для неиндуктивных (слева) и индуктивных (справа) нагрузок. Токоограничивающий резистор R1 может оказаться ненужным. Нагрузка включается, когда входной сигнал от микроконтроллера PIC переходит в состояние высокого уровня

Например, если для управления двигателем требуется 500 мА, то достаточный ток базы составляет всего лишь около 5 мА, что можно легко получить с вывода микроконтроллера PIC.

При выборе транзистора крайне важно убедиться в том, что он способен пропустить ток, необходимый для устройства. Популярный транзистор BC548 пропускает до 100 мА, поэтому он не может использоваться для управления типичным двигателем. Транзистор BC639 пропускает до 1 А, и потому приемлем для управления маленьким двигателем постоянного тока.

На рис. 3.36 показаны два варианта ключа на биполярном плоскостном транзисторе. На схеме справа присутствует диод, включенный параллельно нагрузке. Он называется защитным, поскольку защищает транзистор от риска повреждения при переключении индуктивной нагрузки. К категории индуктивных относят нагрузки, работающие через формирование магнитного поля (например, двигатели, соленоиды и реле). Проблема заключается в том, что при отключении эти устройства индуцируют ток, создающий на транзисторе напряжение в несколько

сотен вольт. Диод же отводит этот ток до того, как он вызовет какие-либо повреждения.

Транзисторный ключ может использоваться для управления устройством, работающем при напряжении, превышающем рабочее напряжение микроконтроллера PIC. Например, если PIC работает от 4,8 В, его выход может управлять двигателем на 12 В (рис. 3.37).

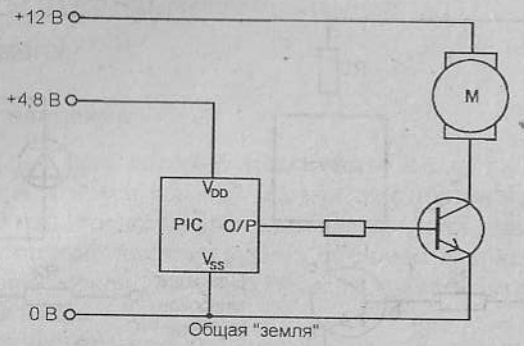



**Рис. 3.37.** Двигатель на 12 В управляется через транзистор от микроконтроллера PIC, работающего от напряжения 4,8 В

Микроконтроллер PIC подключен к источнику питания на 4,8 В, однако коллектор транзистора связан через нагрузку с более высоким напряжением. Обе цепи должны быть связаны с одной и той же линией 0 В.

### *Проектирование ключа на биполярном плоскостном транзисторе*

1) Выберите тип транзистора, который может пропускать требуемый ток и характеризуется достаточным усилением.

2) Когда транзистор полностью открыт, напряжение на его коллекторе составляет приблизительно 0,7 В. Вычислите или определите по спецификации транзистора ток $I_{нагр}$, протекающий через нагрузку. Если он слишком велик для нагрузки, включите в схему последовательный резистор.

3) Падение напряжения на резисторе базы $V_{вых} - 0,7$, где $V_{вых}$ — выходное напряжение высокого логического уровня (в типичном случае $V_{пит} - 0,7$).

4) Минимальный ток, необходимый для полного включения нагрузки ключа, равен $I_{нагр}$ / коэффициент усиления.

5) Сопротивление резистора базы составляет ($V_{пит} - 1,4$) / ($I_{нагр}$ / коэффициент усиления). Используйте резистор E12 с ближайшим номиналом в сторону увеличения.

Ключи, основанные на полевых МОП-транзисторах подобны рассмотренным за исключением того, что ток стока (d) зависит от *напряжения* на затворе (g) (рис. 3.38).



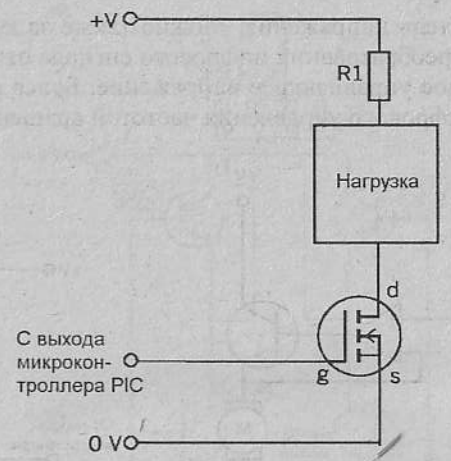



**Рис. 3.38.** Транзисторный ключ на n-канальном полевом МОП-транзисторе. Токоограничивающий резистор может не потребоваться. Если нагрузка носит индуктивный характер, то потребуется защитный диод

Преимущество полевых МОП-транзисторов заключается в том, что они практически не потребляют ток. Их проще включать в схемы, поскольку для них не требуется резистор затвора. Некоторые такие транзисторы характеризуются очень низким сопротивлением проводящего состояния, в силу чего они отдают максимум мощности в нагрузку. Полевые МОП-транзисторы быстрее биполярных плоскостных, однако последние обычно достаточно быстры для схем роботов.

### Управление частотой вращения двигателя

Простейший способ переключения двигателя — с помощью транзисторного ключа. Он полностью включает или полностью отключает двигатель, но не делает ничего другого. Нельзя реверсировать направление или отрегулировать частоту вращения.

Схема, показанная на рис. 3.39, — улучшенный вариант простого ключа. Поскольку механическая нагрузка на двигатель — переменная, частота его вращения — тоже переменная. Это происходит вследствие изменения обратной ЭДС, генерируемой двигателем, что приводит к варьированию напряжения. Операционный усилитель используется для поддержания на двигателе уровня напряжения, равного напряжению управления.

Для обеспечения фиксированной частоты вращения напряжение управления может быть постоянным (скажем, получаемым с фиксированного делителя напряжения). Для переменной частоты используют

переменный делитель напряжения. Можно также задействовать микросхему ЦАП для преобразования цифрового сигнала от микроконтроллера PIC в аналоговое управляющее напряжение. Более простой и эффективный способ цифрового управления частотой вращения иллюстрирует рис. 3.40.

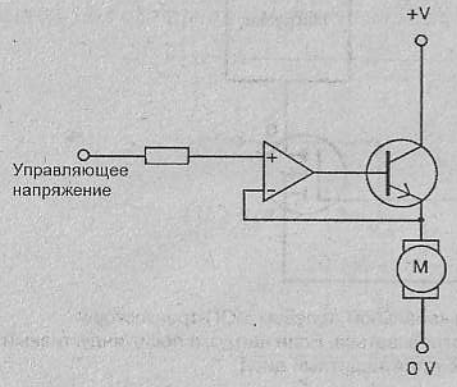



**Рис. 3.39.** Эта схема удерживает частоту вращения двигателя постоянной даже при изменении его механической нагрузки

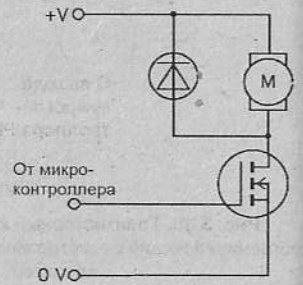



**Рис. 3.40.** Биполярный плоскостной n-p-n-транзистор можно заменить полевым МОП-транзистором

Вместо изменения напряжения данная методика предусматривает питание двигателя фиксированным напряжением, которое, однако, включается только на определенный период времени. Напряжение подается и отключается быстро, поэтому двигатель работает плавно, без рывков.

Частота вращения двигателя управляется импульсным сигналом, генерируемым микроконтроллером PIC. У этого сигнала — переменный коэффициент заполнения (соотношение между длиной импульса и интервалами или промежутками между импульсами). Соответствующая программа представлена в главе 8, "Робот-игрушка".

Преимущество этой методики заключается в том, что на клеммах двигателя — всегда полное напряжение, и он хорошо работает на малых оборотах, без остановок.

### Управление направлением вращения двигателя

Направлением вращения двигателя можно управлять с помощью H-образного транзисторного моста или же с помощью реле. Мостовая схема используется в большинстве проектов, описанных в этой книге. Она может быть выполнена на биполярных плоскостных или полевых МОП-транзисторах (рис. 3.41).



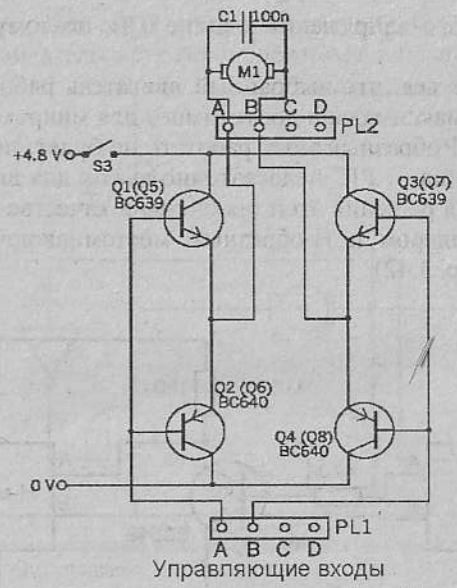



**Рис. 3.41.** Сдвоенная схема управления двигателями в роботе "Искатель". Здесь показана только цепь для левого двигателя. Значения в скобках относятся к цепи правого двигателя

Эта схема состоит из четырех транзисторов для управления одним двигателем. Два из транзисторов — типа n-p-n (Q1 и Q3), а два — типа p-n-p (Q2 и Q4).

Два управляющих входа (A и B) поставляют ток в базы n-p-n-транзисторов и включают их, когда входное напряжение — высокого уровня. Они принимают ток из баз p-n-p-транзисторов и включают их, когда входное напряжение — низкого уровня. Например, если на входе A — напряжение высокого уровня, то транзистор Q1 включен, а Q2 выключен. Если, в то же время, на выходе B — напряжение низкого уровня, то транзистор Q3 выключен, а Q4 включен.

Ток течет от шины положительного напряжения через Q1 и вывод A на двигатель, а через вывод B и Q4 — на шину 0 В. Ток течет через двигатель слева направо на рис. 3.41. Если на выводе A — напряжение низкого уровня, а на выводе B — высокого, то ток течет через двигатель справа налево. Таким образом, H-образный мост действует как реверсирующий ключ.

Если на выводах A и B одновременно — напряжение высокого или низкого уровня, то отсутствует цепь, по которой ток может течь от ши-

ны положительного напряжения к шине 0 В, поэтому двигатель останавливается.

Может случиться, что выбранный двигатель работает от напряжения выше 6,5 В, максимально допустимого для микроконтроллеров PIC. В таком случае Н-образный мост работать не будет, поскольку уровень логической единица от PIC недостаточно высок для выключения p-n-p-транзисторов. Для решения этой проблемы в качестве интерфейса между микроконтроллером и Н-образным мостом включают пару n-p-n-транзисторов (рис. 3.42).

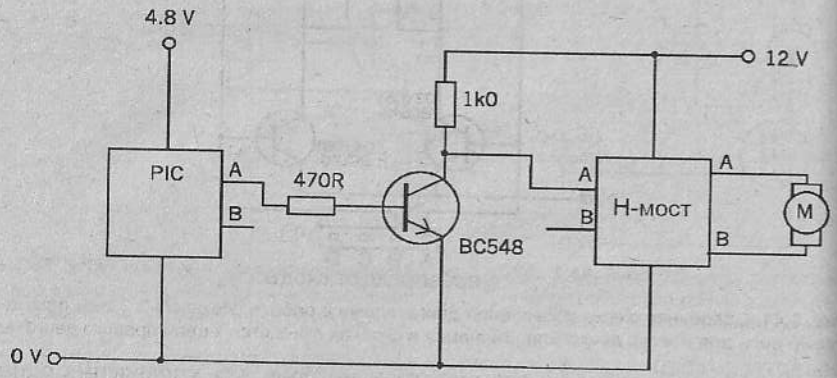



**Рис. 3.42.** Подключение микроконтроллера PIC к Н-образному мосту с более высоким напряжением питания. При этом требуется два транзистора, однако здесь показана только цепь управления для вывода А

Эти транзисторы — с тем же высоким напряжением питания, что и Н-образный мост. Их выходное напряжение в отключенном состоянии близко к напряжению на шине повышенного напряжения, в силу чего транзисторы p-n-p-типа надежно заперты.

Еще одна версия Н-образного моста показана на рис. 3.43. Она построена на четырех n-канальных полевых МОП-транзисторах. Обратите внимание на то, что в этой схеме, в отличие от схемы на биполярных плоскостных транзисторах, все транзисторы — одинаковой полярности. Логические элементы "НЕ" включают транзисторы попарно. Вместо вентилей "НЕ" можно также использовать двухвходовые логические элементы "НЕ-И" или "НЕ-ИЛИ" с объединенными входами.

Если управляющий вход находится в состоянии высокого уровня, то на затворах Q1 и Q4 — низкое напряжение, в силу чего эти транзисторы выключены. На затворах Q2 и Q4 — напряжение высокого уровня, и потому они открыты. Ток течет от положительной шины питания через Q4, двигатель и Q2 на шину 0 В. Если управляющий вход — в состоянии низкого уровня, то Q1 и Q3 открыты, а Q2 и Q4 — выключены. Ток течет через двигатель в противоположном направлении.



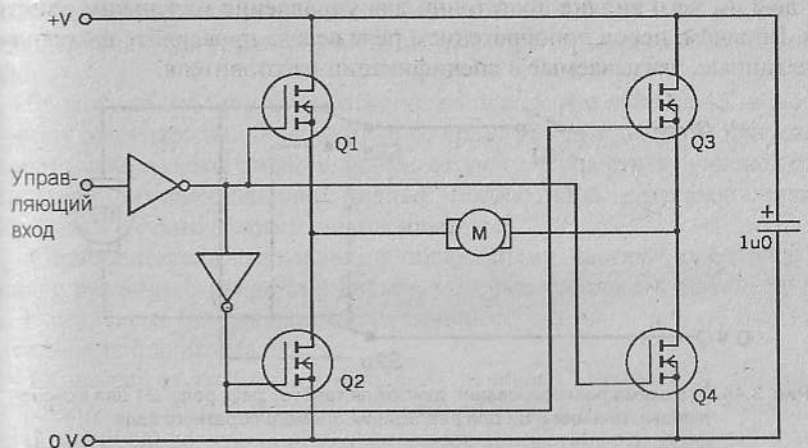



**Рис. 3.43.** Контроллер направления вращения двигателя по схеме Н-образного моста на четырех n-канальных полевых МОП-транзисторах и двух КМОП-вентилях "НЕ"

Схема на рис. 3.43 не обеспечивает никакого способа останова двигателя, однако это можно реализовать при наличии пятого полевого МОП-транзистора, включенного последовательно с мостом. Его сток следует соединить с линией 0 В, а исток — с шиной питания. Напряжение высокого уровня на затворе включает транзистор, и ток течет через мост.

Альтернативой Н-образному мосту как реверсирующему ключу является двухполюсное реле на два направления, обладающее рядом полезных свойств. Переключаемая схема может работать на любом напряжении — как более высоком, так и более низком по отношению к напряжению в цепи микроконтроллера PIC — при условии, что контакты реле рассчитаны на него. Кроме того, существуют реле, выдерживающие ток до нескольких ампер.

Схема, показанная на рис. 3.44, — это типичный реверсирующий ключ, основанный на использовании двухполюсного реле на два направления. Питание на обмотки реле подается через пару транзисторных ключей (рис. 3.45).

Реле, обычно используемые для коммутации двигателей, — устройства для монтажа на печатной плате. Зачастую они имеют такие же габариты, как и 16-выводные микросхемы, иногда — немного больше. Большинство из них рассчитаны на напряжение 12 В, хотя в общем слу-

чае они работают при более низком напряжении. Несколько типов реле рассчитаны на 6 В, другие же — на 5 В. Их контакты могут пропускать ток до 1 А, чего вполне достаточно для управления маленьким двигателем. Впрочем, перед приобретением реле всегда проверяйте их технические данные, указываемые в спецификации изготовителя.

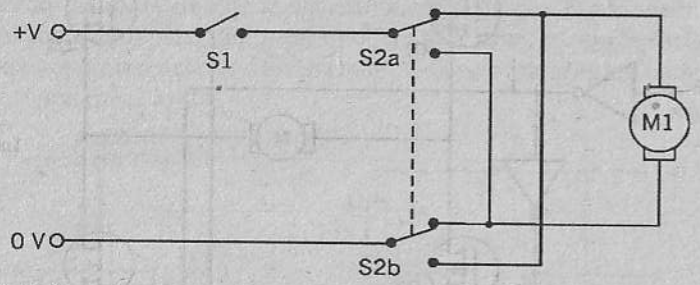



**Рис. 3.44.** Эта схема реверсирования двигателя требует двух реле: S1 для включения/выключения и S2 для реализации прямого/обратного хода

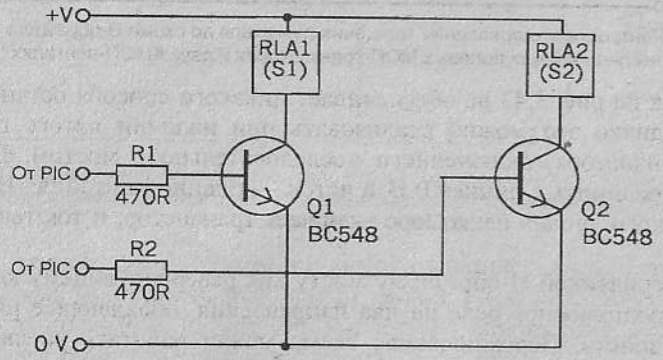



**Рис. 3.45.** Транзисторные ключи используются микроконтроллером PIC для управления схемой реверсирования

### Серводвигатели

Существует несколько типов серводвигателей, отличающихся по размерам, методам управления и временным требованиям. Типичный серводвигатель имеет три клеммы. Положительная клемма и клемма 0 В подключаются к шинам электропитания. Третья клемма обычно связана с генератором импульсов, формирующим последовательность импульсов фиксированной длительности. Обычно длительность импульсов находится в диапазоне 1..2 мс, а частота их следования составляет 50 Гц. Длительность импульсов определяет угол и направление поворота вала.



Так, например, длительность 1 мс приводит к повороту вала влево (против часовой стрелки) до упора. Импульсы длительностью 2 мс заставляют вал поворачиваться вправо (по часовой стрелке) до упора. Импульс средней длительности в 1,5 мс выводит вал в центральную позицию.

Зачастую серводвигатель ограничен поворотом вала на 45° в любую сторону от центральной позиции, в то время как другие типы допускают поворот вала на угол до 90° в любую сторону. Существуют также серводвигатели, обеспечивающие полный оборот вала. Другими словами, сверяйтесь с техническими спецификациями.

Серводвигатели связывают с подвижными частями роботов с помощью различных рычагов и дисков, которые крепятся к выходному валу. Комплекты, содержащие такие приспособления, зачастую поставляются вместе с двигателями.

Серводвигатели удобны в задачах рулевого управления, поскольку их легко установить в позицию "прямо вперед" в начале программы, передав им последовательность импульсов длительности в 1,5 мс в течение 200 мс. Этого времени достаточно для достижения центральной позиции из предыдущей (неизвестной).

### Шаговые двигатели

Как следует из их названия, шаговые двигатели поворачивают вал пошагово. Наиболее распространенный шаг — 7,5°, когда для совершения одного оборота требуется 24 шага. Такие двигатели дают немного толчкообразное движение, которое, впрочем, вполне приемлемо для большинства целей. При этом их просто программировать. Представленное ниже описание относится к униполярным шаговым двигателям с постоянными магнитами (шаг 7,5°).

Такой шаговый двигатель обычно имеет шесть проводов, подключенных к обмоткам согласно рис. 3.46. Два отвода подключаются к положительной линии электропитания. Концы обмоток соединены с одним из четырех транзисторных ключей, которые попеременно включаются и отключаются. Когда ключ выключен, конец обмотки подключен к положительной шине питания, и в соответствующей половине обмотки ток не протекает. Когда ключ включен, напряжение на конце обмотки падает почти до нуля, и на соответствующую половину обмотки подается питание.

Транзисторы на двух концах обмотки переключаются противоположным образом, поэтому питание всегда подается только на одну половину обмотки, и двигатель поворачивает вал на один шаг (рис. 3.47).

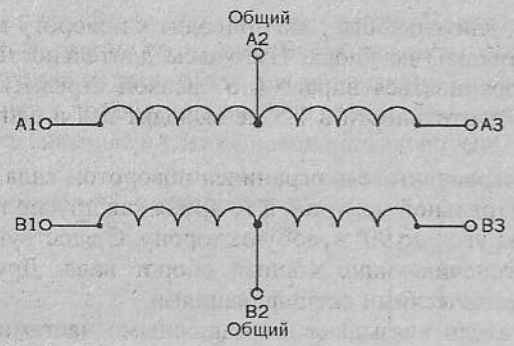



**Рис. 3.46.** Две обмотки шагового двигателя имеют отводы от центра

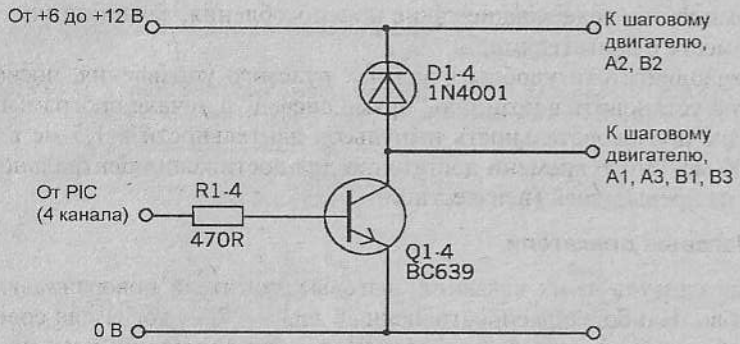



**Рис. 3.47.** Цепь управления шаговым двигателем. Требуется четыре подобных ключа

Двигатель управляется путем включения транзисторов в соответствии со специальной последовательностью (0 = выкл.; 1 = вкл.):

- Q1 — 1 1 0 0;
- Q2 — 0 0 1 1;
- Q3 — 0 1 1 0;
- Q4 — 1 0 0 1.

Транзисторы включаются и выключаются один раз в течение каждой последовательности, однако на различных этапах. Для управления двигателем микроконтроллер PIC программируется на формирование последовательности единичных импульсов в четырех выходных цифровых каналах (обычно — четыре канала одного и того же порта). Последовательность повторяется до тех пор, пока это необходимо. Каждый шаг дает поворот на 7,5°, а последовательность содержит четыре этапа, в силу чего одна полная последовательность поворачивает вал на 30°. Таким образом, одному обороту вала соответствует двенадцатикратное



прохождение последовательности. Последовательность может быть остановлена на любом шаге, следовательно, точность позиционного управления двигателем составляет 7,5°.

Когда последовательность импульсов прекращается, двигатель останавливается. Однако два транзистора все еще включены, и ток течет в двух полуобмотках. Вал твердо удерживается в неподвижном состоянии магнитными полями. Удерживающий момент является — большой, порядка нескольких сотен грамм-сантиметров.

Частота вращения двигателя контролируется частотой следования последовательности импульсов. Реверсировать шаговый двигатель — просто. Для этого последовательность импульсов следует пройти в обратном порядке.

Программирование шагового двигателя описано в главе 7, "Андроид".

## Радиосвязь

Два робота, которые могут "переговариваться" и взаимодействовать друг с другом открывают захватывающие возможности. Очевидный способ общения для них — радиосвязь, использующая модули передатчика и приемника, которые работают на частоте 433,92 МГц (рис. 3.48). Эти модули массово производятся для использования в устройствах с дистанционным управлением, автомобильных системах сигнализации и т.п., так что их стоимость минимальна. Кроме того, поскольку они рассчитаны на переносные устройства, они работают при малых напряжениях питания.

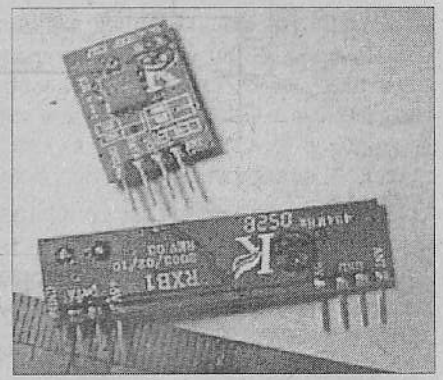


**Рис. 3.48.** Передатчик и приемник — эффективный способ организации взаимодействия двух роботов или дистанционного управления человеком. Двухсторонняя связь требует по два таких модуля

Существует несколько типов радиомодулей, однако многие из них подобны описанным ниже. В случае каких-либо сомнений обращайтесь к техническим спецификациям.

Схемы, показанные на рис. 3.49 и рис. 3.50, демонстрируют, насколько просто модули передатчика и приемника используются совместно с микроконтроллерами PIC.

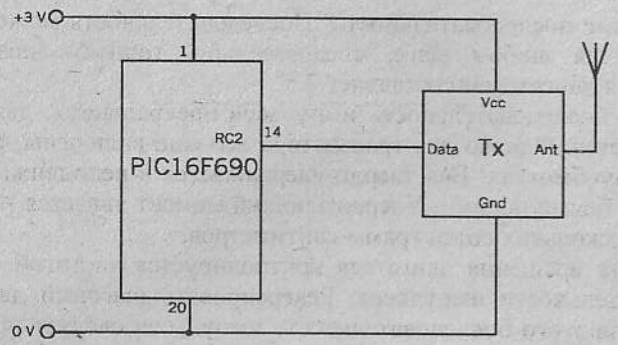



**Рис. 3.49.** Цифровой сигнал из микроконтроллера PIC подается непосредственно на вывод Data (данные) модуля передатчика. В данном случае к выводу Ant подключена антенна, однако на расстояниях до нескольких метров модуль работает хорошо и без нее

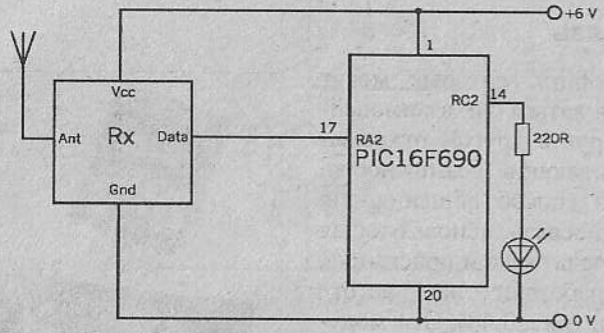



**Рис. 3.50.** Сигнал с вывода Data подается непосредственно на цифровой входной канал микроконтроллера PIC, после чего дублируется на светодиод. Опять-таки, для коротких расстояний можно обойтись без антенны. Обратите внимание на более высокое рабочее напряжение модуля приемника

Эти модули потребляют всего лишь несколько миллиампер тока. На рис. 3.49 и рис. 3.50 микроконтроллер PIC работает на том же питающем напряжении, что и передатчик с приемником. Поскольку передатчик работает на более низком напряжении, напряжение питания PIC, равное 6 В, делится, чтобы предоставить 3 В для передатчика, когда система работает с двумя модулями. Выходной сигнал микроконтроллера подается на передатчик через делитель напряжения.

## Тестирование схем

Разработайте процедуру тестирования схем для каждой стадии конструирования. Затем протестируйте всю систему в целом, чтобы убедиться, что все ее части работают вместе как должно. Ниже представлен общий шаблон такой процедуры. Детали зависят от вида тестируемой схемы.



1. **Визуальный контроль**. На данном этапе микроконтроллер PIC в колодку не установлен. Питающее напряжение не подается. Проверьте корректность соединений на печатной плате, номиналов резисторов и конденсаторов, установки диодов и электролитических конденсаторов, размещения разрезов на медных полосках (используйте лупу), а также убедитесь в отсутствии капель припоя или волос, которые могут вызвать короткое замыкание (опять-таки, используйте лупу).
2. **Неразрывность**. На данном этапе микроконтроллер PIC в колодку не установлен. Питающее напряжение не подается. С помощью средств контроля неразрывности мультиметра убедитесь в непрерывности линии 0 В, линии (линий) положительного напряжения, а также любых соединений со внешними компонентами.
3. **Короткозамкнутые цепи**. На данном этапе микроконтроллер PIC в колодку не установлен. Питающее напряжение не подается. С помощью средств контроля неразрывности мультиметра убедитесь в отсутствии короткого замыкания между линией 0 В и линией (линиями) положительного напряжения, а также — между соседними выводами гнезд для микросхем, включая микроконтроллер PIC.
4. **Подача электропитания**. На данном этапе микроконтроллер PIC в колодку не установлен. Подсоедините отрицательный вывод мультиметра к клемме 0 В батареи или блока питания, и переключитесь в режим измерения напряжения. Подайте питание и проверьте все точки, в которых должно присутствовать напряжение питания. Если в какой либо точке напряжение окажется низким, немедленно отключите питание и поищите короткозамкнутую цепь, резисторы с неправильным номиналом или нерабочие компоненты.
5. **Выходы**. На данном этапе микроконтроллер PIC в колодку не установлен. Подсоедините гибкие проводники к линии 0 В и линии положительного напряжения. Касайтесь ими отдельных выходов в гнезде установки микроконтроллера PIC, чтобы убедиться в том, что они работают правильно (т.е. светодиоды светятся, двигатели вращаются и т.д.).
6. **Входы**. На данном этапе микроконтроллер PIC в колодку не установлен. С помощью мультиметра в режиме измерения напряжения измерьте напряжения на выходах в гнезде для установки микрокон-

троллера PIC. Замкните входные (микро)переключатели и убедитесь, что входные напряжения изменяются корректно. Затеняйте или освещайте фотоэлементы, контролируя соответствующие напряжения.

7. **Тестирование микроконтроллера PIC**. Отключите напряжение питания, установите запрограммированный микроконтроллер PIC в его колодку, и опять подайте питание. Для некоторых проектов мы создали простые диагностические программы, которые проверяют систему по частям. Это упрощает поиск ошибок в системе или в программах. В случае с мобильным роботом его необходимо зафиксировать на опорах, чтобы колеса висели воздуха. Это предотвратит "побег" робота во время тестирования.

## Ссылки в Internet

Электронные компоненты, упомянутые в этой части, можно получить от поставщиков, работающих на рынке товаров для радиолюбителей. Почти все они поддерживают поставку по почте. Перечислим некоторые из полезных Web-сайтов:

- `maplin.co.uk` — известная английская компания, поставляющая товары по всему миру;
- `newark.com` — поставляет широкий диапазон электронных компонентов в США и по всему миру;
- `crownhill.co.uk` — материалы по PIC, PICBASIC, макетным платам и др.;
- `www.microchip.com` — изготовитель микроконтроллеров PIC;
- `imagesco.com` — поставляют робототехнические и электронные компоненты. Сайт предоставляет неплохой выбор датчиков (включая недорогие магнитные компасы), статьи по разработке роботов с программами на PICBASIC. Продается программное обеспечение PICBASIC.

Поставщиков гораздо больше, чем можно здесь перечислить. Для их поиска используйте средства, наподобие Google и Yahoo. Например, мы нашли горы информации по таким ключевым словам и фразам, как "реле", "шаговый двигатель" и "фототранзистор".

# Часть II

# Микроконтроллеры PIC

***В этой части...***

- Использование микроконтроллеров PIC
- Программирование микроконтроллеров PIC

# Глава 4. Использование микроконтроллеров PIC

Робототехнические проекты, описанные в этой книге, построены на микроконтроллере PIC16F690 от компании Microchip Technology Inc., однако все микроконтроллеры PIC обладают общей базовой архитектурой и схожими электрическими характеристиками, так что рассмотренные схемы, как правило, применимы и для некоторых других типов PIC.

## Программирование микроконтроллеров PIC

Микроконтроллер PIC16F690 особенно хорош для любительских робототехнических проектов, потому что:

- это один из наиболее новых представителей семейства PIC;
- если синхронизация реализована с помощью встроенного осциллятора, то все 20 выводов, кроме двух, будут доступны для использования в качестве цифровых входов и выходов, а значит робот можно оснастить большим числом датчиков и исполнительных механизмов;
- оснащен флэш-памятью, идеальной для программирования любительских роботов;
- память программ на 4 096 слов (14-разрядных), что в четыре раза больше, чем у PIC16F84 (предыдущего наиболее популярного микроконтроллера PIC); 256 байт памяти SRAM и 256 байт памяти EPROM;
- широкий ассортимент встроенных устройств типа генераторов, таймеров, аналоговых компараторов и аналого-цифровых преобразователей, что очень полезно при сборке усложненных роботов;
- один из наиболее дешевых микроконтроллеров PIC;
- программируется на новейшем программаторе PICkit 2 и может быть протестирован на демонстрационной плате Microchip;
- из-за сходства архитектур все микроконтроллеры PIC среднего подсемейства (например, F690) используют один и тот же ассемблер из 35 команд;



- использует новейшую технологию nanoWatt. Ток режима ожидания для PIC16F690 составляет всего лишь 1 нА при рабочем напряжении 2 В. При максимальном напряжении питания 5,5 В и частоте 4 МГц рабочий ток составляет менее 1 мА. Низкие напряжения и токи — идеальный вариант для управления мобильными роботами.

Предполагая, что читатель уже немного знаком с программированием на ассемблере (на начальном уровне), в этой главе будут рассмотрены вопросы, которые не всегда включают в упрощенные руководства по программированию — особенно в области робототехники.

### Аппаратные средства для программирования

Для программ, рассмотренных в этой книге, мы использовали программатор PICkit 2 (рис. 4.1), который можно получить по почтовому заказу от нескольких поставщиков, включая непосредственно компанию Microchip (`www. microchip.com`).

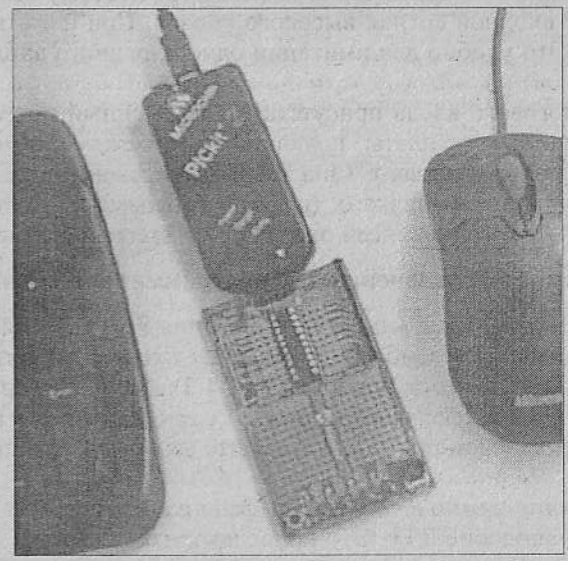



**Рис. 4.1.** Программатор PICkit 2 оснащен USB-разъемом. Этот небольшой модуль не занимает много места на компьютеризованном рабочем месте при программировании и тестировании PIC. Он соединяется непосредственно с маленькой платой, в которую устанавливается микроконтроллер на время его программирования. Плата оснащена переключателем и несколькими светодиодами для тестирования подпрограмм ввода-вывода. Она также позволяет формировать различные интерфейсы с микроконтроллером

Этот сайт предоставляет всю необходимую информацию о мире PIC. Посетите его и просмотрите техническое описание микроконтроллера PIC16F690. Здесь же присутствует ссылка на сопутствующий сайт Microchip Direct, позволяющий заказывать микроконтроллеры PIC, программатор PICkit 2 и другие аппаратные средства по почте.

### Демонстрационная плата

Демонстрационная плата PICkit 2 Low Pin Count оснащена четырьмя светодиодами, управляемыми выходными каналами RC0..RC3. Они очень полезны при отладке программ, поскольку отображают состояние выходов. Если программа должна выдавать сигналы через другие каналы, то на время разработки программы их можно перенаправить на RC0..RC4, и изменить целевые выходы позже, после того как программа будет отлажена.

В левом переднем углу демонстрационной платы также размещен кнопочный переключатель SW1, связанный с каналом RA4. Данный вывод имеет внешний подтягивающий резистор, поэтому на нем обычно присутствует входной сигнал высокого уровня. При нажатии SW1 уровень падает. Это удобно для имитации одноразрядного входного датчика.

Для аналогового ввода присутствует переменный потенциометр на переднем правом крае платы, подключенный между линиями питания и 0 В. Его ползунок связан с каналом RA1. Уровень входного сигнала может изменяться от низкого (крайнее положение против часовой стрелки) до высокого (крайнее положение по часовой стрелке).

### Программное обеспечение для программирования

Наиболее прямой путь программирования PIC заключается в написании программы на ассемблере с помощью простого текстового редактора, наподобие Блокнота Windows (рис. 4.2). Файл, созданный в Блокноте, сохраняется в текстовом формате `.txt`. Когда редактор запросит ввести имя сохраняемого файла, укажите для него расширение `.asm` вместо `.txt`.

Существенно важно *не* сохранить файл с ассемблерной программой в форматах, наподобие RTF. Это приведет к добавлению в текст управляющих кодов форматирования, которые запутают ассемблер на следующем этапе.

Плата программатора обычно поставляется вместе с программным обеспечением, которое позволяет преобразовать текстовый файл в машинные коды, загружаемые в микроконтроллер PIC. В пакет поставки



PICkit 2 входит оболочка MPASM. Программа ассемблера, называемая MPASMWIN, показана на рис. 4.3.

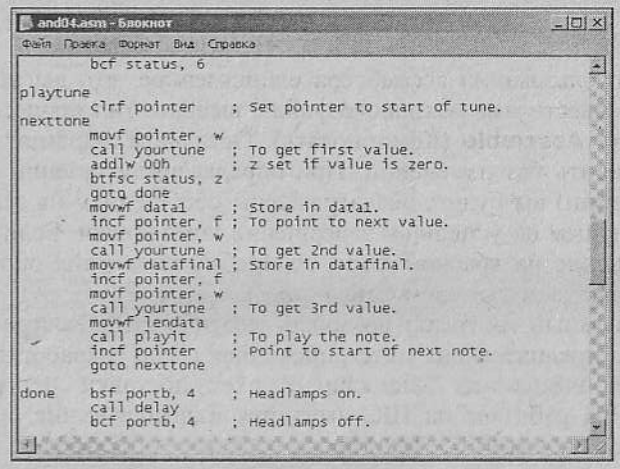



**Рис. 4.2.** Ассемблерную программу можно написать с помощью текстового редактора типа Блокнот. Здесь мы видим начало программы для робота, рассмотренного в главе 7. У левого края командных строк расположены метки. Команды размещают с отступом, устанавливаемым табулятором. Комментарии находятся справа после точки с запятой. Достаточное количество комментариев помогает анализировать программу в будущем

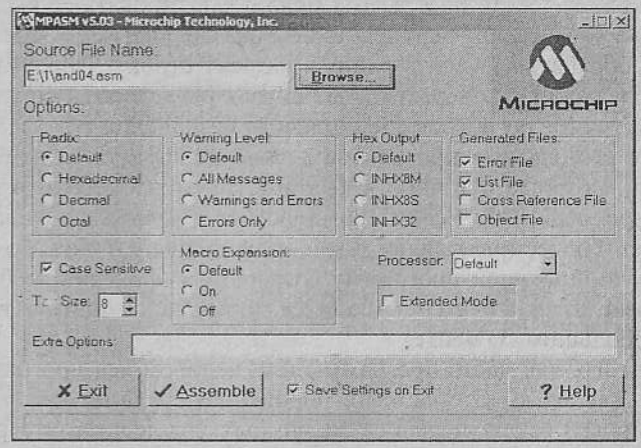



**Рис. 4.3.** Окно MPASM позволяет установить различные параметры создания файла машинного кода (файл `.hex`). В ходе компоновки выполняется поиск ошибок, и выдаются соответствующие сообщения и предупреждения. При наличии ошибок отчет о них сохраняется в файле `.err`, который можно просмотреть с помощью того же Блокнота

**Windows**

Программы, рассмотренные в этой книге, были созданы на ПК, работающем под управлением Windows XP. Использованное программное обеспечение также работает под Windows Vista (версия Home Basic).

При использовании ассемблера единственное, что вы необходимо сделать, — ввести имя исходного файла, выбрать тип процессора и нажать кнопку **Assemble** (Компоновать). Остальные параметры обычно можно оставить без изменений. При определенном везении (или, возможно, умении) вы будете вознаграждены сообщением на зеленом фоне, сообщающем об успешном завершении компоновки. Если будет выдано сообщение на красном фоне, то просмотрите файл ошибок `.err` (его имя совпадаем с именем файла `.asm`).

Программный имитатор позволяет достичь цели быстрее. Именно для этого и предназначена интегрированная среда разработки MPLAB, включенная компанией Microchip в пакет поставки программатора PICkit 2. Она работает на ПК, имитируя взаимодействие с реальным микроконтроллером PIC. Текстовый редактор MPLAB предоставляет различные возможности форматирования, включая отображение элементов листинга несколькими цветами: метки — красные; команды — синие, выделенные полужирным шрифтом и т.д. Благодаря этому, листинг более читабелен и прост для восприятия.

После ввода текста программы можно имитировать ее выполнение. Так, на рис. 4.4 в левом верхнем окне показан листинг. Стрелка слева указывает обрабатываемую в данный момент строку. Внизу находится окно, отображающее содержимое файловых регистров, а справа можно увидеть состояние регистров специального назначения.

В ходе выполнения программы можно наблюдать за изменениями в регистрах, которые в точности соответствуют процессам в реальном микроконтроллере PIC. Если некоторую часть программы необходимо исследовать более пристально, то ее можно пройти пошагово. Будет меньше путаницы, если программу создавать и проверять небольшими фрагментами. Если что- то не заладится, то проще локализовать ошибку в новом небольшом сегменте.

После того, как часть программы или целиком вся программа заработает так, как это требуется, интегрированная среда вызывает средства программирования для переноса машинных кодов в память реального микроконтроллера PIC с помощью программатора. После этого разместите микроконтроллер в гнезде на роботе, и в добрый путь!

Мы не сможем в рамках этой книги описать все полезные функции MPLAB. Просто загрузите эту среду и поработайте с ней сами.



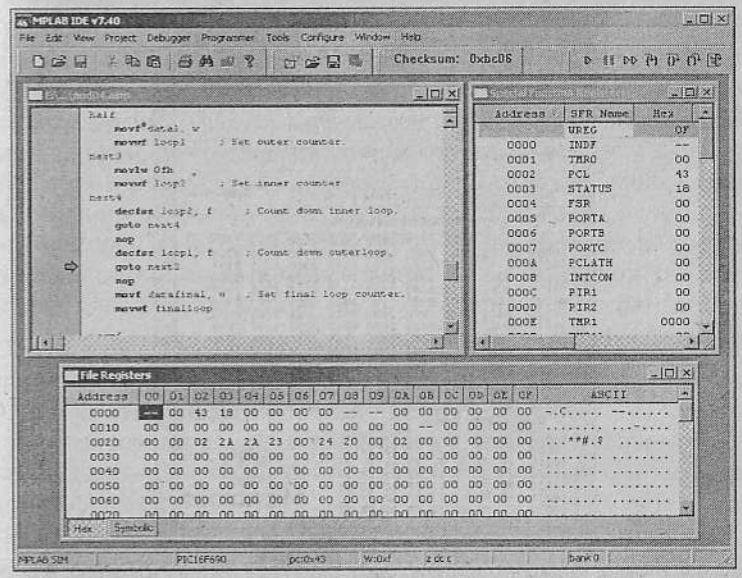



Рис. 4.4. Интегрированная среда MPLAB имитирует работу микроконтроллера PIC16F690

В пакет поставки PICkit 2 входит программное обеспечение, которое может использовать независимо от имитатора. При желании для написания ассемблерной программы можно использовать Блокнот, для ее компоновки в файл `.hex` — MPASM, а для загрузки этого файла в микроконтроллер PIC — программу PICkit 2 (рис. 4.5). Цель этой главы — дать читателю представление о методиках программирования PIC. На рынке присутствуют и другие редакторы и программаторы, и их ассортимент постоянно расширяется. Кроме того, можно использовать программное обеспечение, работающее с языками BASIC и C.

### Языки программирования высокого уровня

Ассемблер инструктирует микроконтроллер шаг за шагом, языки же высокого уровня, наподобие BASIC и C, предоставляют команды, каждая из которых определяет действия микроконтроллера в нескольких шагах. Это ускоряет и упрощает программирование. Хороший пример — команда `WRITE` в PICBASIC. Так, одна строка программы

```
WRITE 3, count
```

помещает значение переменной `count` в байт 3 памяти EEPROM микроконтроллера PIC. Для реализации этого же действия на ассемблере потребует намного больше программных строк.

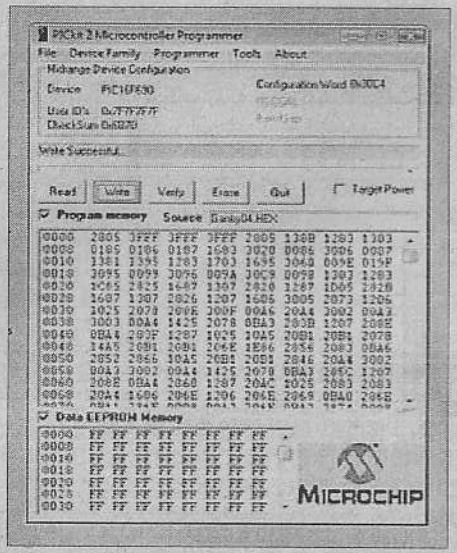



**Рис. 4.5.** PICkit 2 только что загрузил файл `Gantry04.hex` в PIC16F690. В окне показаны шестнадцатеричные коды

Листинг программы, написанной на языке высокого уровня, обычно короче, в силу чего на его ввод и проверку уходит меньше времени. За это приходится платить тем, что результирующий машинный код зачастую длиннее, чем в случае эквивалентной ассемблерной программы. Он требует для своего размещения больше памяти, а на выполнение такой программы уходит больше времени. Таким образом, ассемблер выигрывает по компактности и скорости выполнения, язык же высокого уровня более прост в изучении и понимании.

Наиболее популярные языки высокого уровня сопровождаются соответствующей интегрированной средой разработки, которая работает на ПК и состоит из ряда модулей (текстовый редактор для ввода программ; транслятор, преобразующий эти программы в машинный код; отладчик для тестирования кодов на моделируемом PIC; программное обеспечение для передачи машинных кодов в память программ микроконтроллера).

В настоящее время доступны следующие интегрированные среды разработки:

- Proton — `www.mecanique.co.uk`;
- SourceBoost — `www.sourceboost.com`;
- Microcode Studio — `www.melabs.com`.



Они подобны по своей общей структуре и по набору предоставляемых средств. Каждая из них использует собственную версию BASIC:

- Microcode Studio — фактически PICBASIC, который очень похож на стандартный BASIC, но реализует дополнения, направленные на адаптацию к программированию микроконтроллеров PIC;
- Proton — здесь BASIC подобен предыдущему, однако включает в себя намного больше специальных команд;
- SourceBoost — наиболее выделяется среди названных трех версий BASIC, поскольку реализует много свойств, общих с языками C и Java. Он ориентирован на профессиональных программистов.

Тем, кто склонен к минимализму и не хочет использовать интегрированную среду, может установить PICBASIC (с сайта microEngineering Labs). Для ввода и редактирования программ используется простой текстовый редактор типа Блокнота Windows. Можно использовать также и другие редакторы при условии, что они сохраняют файлы в простом текстовом формате.

Введите программу и сохраните ее в файле `.txt`. Затем (если не используется интегрированная среда разработки) активизируйте режим эмуляции DOS. Для этого можно выбрать в системном меню **Пуск** команду **Выполнить** и выполнить с помощью диалогового окна **Запуск программы** команду `cmd`. На будущее все эти действия можно свести к одной пиктограмме, размещаемой на Рабочем столе.

В самой нижней строка в командном окне указано приглашение `C:\<некоторая папка>`. Введите `CD` `..` и нажмите клавишу <Enter>. Повторяйте эти действия до тех пор, пока в строке приглашения не останется только `C:\`, после чего введите `CD PBC` и нажмите клавишу <Enter>. Тем самым была задана папка с компилятором PICBASIC.

Для вызова компилятора введите `PBC -p16F90 -qtxt`, и после пробела — имя файла программы на BASIC (включая расширение `.txt`). Опция `-p` указывает компилятору на модель программируемого микроконтроллера PIC. Опция `-q` дает указание загружать файл с расширением `.txt`. Этот процесс объясняется более подробно в руководстве по использованию языка программирования PCBASIC.

Для того чтобы откомпилировать программу, просто нажмите клавишу <Enter>. После секундной паузы появится сообщение о количестве слов в скомпилированном файле (рис. 4.6). При наличии ошибок в текстовом файле они будут перечислены на экране с указанием соответствующих номеров строк. При подсчете строк учитывайте пустые. Устранив ошибки, выполните повторную компиляцию.


**Рис. 4.6.** Командное окно DOS со строкой вызова компилятора и сообщением об успешной компиляции программы

Окончательный, свободный от ошибок файл программы теперь будет находиться в той же папке и иметь то же имя, но с расширением `.hex`. Он готов к загрузке в PICkit 2 или другой программатор, как это было показано выше.

### Блок-схемы алгоритмов

Микроконтроллеры по своей природе работают маленькими шагами. Они выполняют длинные последовательности простых операций. Это означает, что иногда трудно понять, какой фрагмент программы выполняется, и в правильном ли направлении движется процесс. Блок-схема дает общее представление о программе или ее части (рис. 4.7).

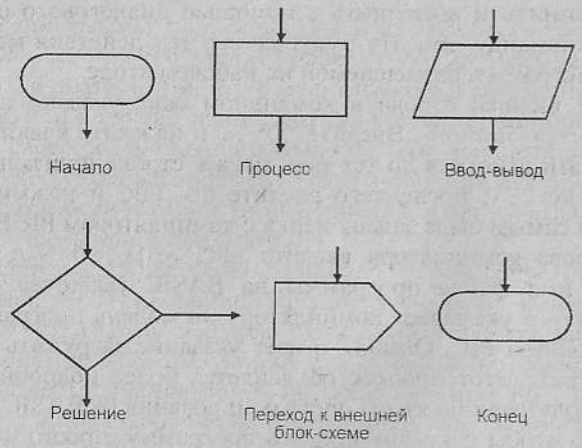



**Рис. 4.7.** Символы, используемые в блок-схемах в этой книге

Блок-схемы полезны на стадии планирования для распределения основных этапов программы. Блок-схема также может оказаться полезной и по завершении разработки программы, поскольку позволяет определить, действительно ли программа работает именно так, как должна.



### Формальный прогон

Какое бы изощренное ПО программирования ни использовалось, иногда единственный реальный способ отладки программы заключается в ее формальном прогоне. Все, что для этого необходимо, — карандаш и бумага.

В начале формального прогона создается таблица всех задействованных регистров и переменных. Затем следует двигаться по листингу строка за строкой, определяя значения переменных в каждом регистре. Вносите их в таблицу и проверяйте корректность их значений. В качестве примера ниже показан фрагмент листинга для части подпрограммы из главы 9.

```
Листинг          Содержимое регистров
                 bitn      bitm      randval   w         c
clrf bitn        00000000  xxxxxxxx  11011011  xxxxxxxx  x
clrf bitm                  00000000
btfsc randval, 5                     (бит 5 = 0)
bsf bitn, 0            (пропуск)
btfsc randval, 6                     (бит 6 = 1)
bsf bitm, 1                00000001
movf bitn w                                    00000000
xorwf, bitm, w                                 00000001
addlw '00FF'                                   00000000  1
rlf randval, f             10110111

c - это разряд переноса регистра STATUS.
Значение x - произвольное
```

Данная подпрограмма формирует новое псевдо-случайное число в `randval` при каждом запуске. Выполните эту подпрограмму на бумаге несколько раз, чтобы убедиться, что формируемые ею значения действительно кажутся случайными.

### Диагностическое программирование

Полезные индикаторы текущего состояния микроконтроллера PIC — это светодиоды. Предположим, мы подозреваем, что PIC не переходит в некоторую подпрограмму, хотя должен это делать. Добавьте в начало подпрограммы строку вида `bsf portb, 2` (или какой-либо другой выходной канал, питающий светодиод). События после запуска программы покажут, где искать ошибку. Если светодиод не светится, то это означает, что подпрограмма не вызывается. Анализируйте листинг, чтобы найти причину отсутствия вызова. Если же светодиод светится, то подпрограмма вызывается, и ошибка — внутри нее. Найдя ошибку,

исправьте ее, удалите дополнительную строку и выполните повторную компоновку.

## Микроконтроллер PIC16F690

Программирование микроконтроллеров PIC, главным образом, заключается в записи данных в массивы регистров. Доступны два вида регистров:

- **регистры специального назначения** — содержат данные, имеющие отношение к входным и выходным портам, компараторам, таймерам, другим функциям, а также — общему управлению микроконтроллером (подробнее рассматриваются далее);
- **регистры общего назначения** хранят обрабатываемые программой данные.

Данные в регистрах хранятся в виде байтов и при выключении питания теряются. В регистрах специального назначения данные принимают исходные значения по умолчанию при подаче питания. Зачастую значения по умолчанию — как раз то, что требуется, а доступ к этим регистрам необходим только для установки другого значения или чтения данных.

### Выводы и порты

Рассматриваемая здесь модель F690 выполнена в плоском корпусе с 20-ю выводами с двухрядным расположением (рис. 4.8). Существуют и другие варианты корпусов (например, для поверхностного монтажа).

Все выводы, за исключением 1 и 20, могут быть использованы для ввода-вывода. Выводы портов A и B могут индивидуально настраиваться на использование встроенных подтягивающих резисторов при их конфигурировании

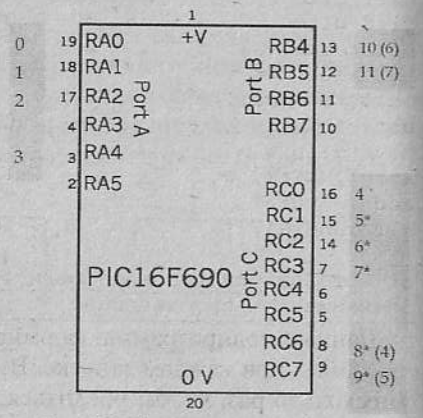



**Рис. 4.8.** Номера портов и выводов микроконтроллера PIC16F690. Каналы каждого порта пронумерованы от 0 до 7 от наименее к наиболее значащему разряду. Порт A предоставляет только шесть каналов, а порт B — каналы только для четырех старших разрядов. Числа на сером фоне — это цифровые входные каналы от AN0 до AN11. Для диапазона от AN0 до AN7 также указаны номера разрядов в регистре ANSEL. Для диапазона от AN8 до AN11 в скобках указаны номера разрядов в регистре ANSELH. Каналы, отмеченные звездочкой, также могут быть использованы как входы компараторов



в качестве входов. Кроме того, в качестве входов они могут программироваться на инициализацию прерываний при изменении входных сигналов.

Некоторые выводы, хотя и могут быть использованы для простого ввода-вывода, могут программироваться на выполнение специальных функций. Например, вывод 4 (RA3) можно запрограммировать как вход общего сброса (при низком уровне сигнала на этом выводе микроконтроллер PIC начинает выполнять программу заново).

Еще один пример — вывод 10 (RB7). В случае использования приемопередатчика USART (рассматривается ниже) этот вывод необходимо задействовать для передачи данных в другой микроконтроллер PIC по физической или радиолинии. Более подробно этот особый вывод рассматривается чуть позже.

### Разряды

Во всех описаниях в этой книге используется следующая терминология. Восемь разрядов байта нумеруются справа (разряд 0, младший, LSB) налево (разряд 7, старший, MSB).

Ссылаясь на некоторый разряд регистра, мы указываем имя регистра, а после него — номер разряда в угловых скобках. Например, разряд RB5 (разряд 5 регистра PORTB) обозначается как RB<5>. Это же соглашение используется во всех спецификациях микроконтроллеров PIC.

Диапазон разрядов обозначается путем указания его старшего и младшего разрядов (именно в таком порядке!). Например, младшие четыре разряда PORTB — это RB<3:0>.

### Регистры специального назначения

Эти регистры контролируют все аспекты работы микроконтроллера PIC. Операции записи/чтения данных для них выполняются побайтно или с отдельными разрядами. В некоторых случаях разряды предназначены только для чтения или только для записи. Регистры специального назначения вместе с восьмиразрядным **рабочим регистром** (обозначается как `w`) формируют рабочую среду.

Регистры специального назначения размещены в четырех областях памяти (банки от 0 до 3), причем в каждый момент времени можно осуществлять доступ только к одной из таких областей. Некоторые регистра (например, STATUS) находятся в одной и той же позиции во всех четырех банках, другие же входят в состав только одного или двух банков. Например, регистр TRISA (определяет, какие разряды порта A будут входами, а какие — выходами) присутствует только в банках 0 и 3.

Банк, доступный в данный момент, определяется значениями разрядов 5 и 6 регистра STATUS. Например, для выбора банка 2 следует установить разряды <6:5> в двоичное 10 (десятичное 2). Контроль текущего банка — крайне важный элемент программирования. В обычном случае используется банк 0, содержащий наиболее популярные регистры, наподобие STATUS и регистров портов.

Рассмотрим пример переключения банков. Первая задача большинства программ — инициализация входов и выходов. После подачи питания все выводы являются входами, поэтому для того, чтобы, например, включить светодиод, мы должны сделать соответствующий вывод выходом. Светодиод может быть подключен между выводом 16 и линией 0 В. Вывод 16 — это RC0, соответствующий разряду 0 порта C.

Предположим, мы работаем с банком 0. При этом STATUS<6:5> = 00. Для того, чтобы сделать RC0 выходом, мы переключаемся в банк 1 путем установки разряда STATUS<5>. Затем мы делаем RC0 выходом, обнулив соответствующий разряд в регистре TRISC. После этого мы возвращаемся в банк 0 путем обнуления разряда 5 в регистре STATUS.

Фрагмент программы, выполняющий перечисленные операции, выглядит следующим образом:

```
bsf status, 5       ; Банк 1.
bsf trisc, 0        ; Делаем RC0 выходом
bcf status, 5       ; Возвращаемся в банк 0
```

Множество других примеров переключения банков представлено в программах, рассмотренных в этой главе, а также — в части III книги.

### Конфигурационное слово

Это 14-разрядное слово, которое хранится по специальному адресу в памяти, определяет характер работы микроконтроллера. Его определение — одна из первостатейных задач программы. Не вдаваясь в детали ряда функций, отметим, что конфигурационное слово, используемое для программ в этой книге, имеет следующую структуру:

- <13:12> — не реализовано, устанавливается в 11;
- <11:6> — разрешение или запрет функций для максимально простой работы (устанавливайте в 000011);
- <5> — делает вывод 3 (RA4) цифровым входом (устанавливайте в 0);
- <4> — включает таймер подачи питания (устанавливайте в 0);
- <3> — отключает сторожевой таймер (устанавливайте в 0);
- <2:0> — выбирает INTOSCIO, внутренний осциллятор с выводом 2 (RA5) для обычного ввода-вывода, а не выхода осциллятора (устанавливайте в 100).



В совокупности эти параметры дают значение 11 0000 1100 0100, что в шестнадцатеричном виде выглядит как 30C4.

Использование других настроек дает незначительное различие в характере выполнения программ, однако в наших программах мы принимаем, что разряд <5> установлен в 0, и выбран внутренний осциллятор.

### Порты

Порты соединяют микроконтроллер PIC со внешним миром. Каждому порту соответствует регистр специального назначения в банках памяти 0 и 2: PORTA, PORTB и PORTC. Хотя все эти регистры — байтовые, в портах A и B реализованы не все каналы (см. рис. 4.8.).

Одинаковые адреса в банках памяти 1 и 3 соответствуют регистрам трех состояний трех портов: TRISA, TRISB и TRISC. Эти регистры определяют, являются ли каналы входными или выходными. Установка соответствующего разряда в 1 делает канал входным, а в 0 — выходным.

При подаче питания все каналы устанавливаются как входы, поэтому, если их использование предполагается именно в такой конфигурации, то никаких действий предпринимать не требуется. Однако в микроконтроллере F690 каналы автоматически устанавливаются как *аналоговые* входы. Способ конфигурирования их в качестве цифровых входов будет рассмотрен в следующей главе.

## Специальные функции

### Компараторы

Микроконтроллер 16F690 содержит два компаратора напряжения, каждый — с собственной схемой управления и конфигурационным регистром. В случае компаратора 1 (C1) этот регистр называется CM1CON0 и расположен по адресу 119h в банке 2. Мы рассмотрим только программирование C1, поскольку компаратор C2 программируется аналогично.

Свойства компаратора микроконтроллера типично для схем сравнения. У него — два входа: C1VP (положительный или неинвертирующий) и C1VN (отрицательный или инвертирующий вход). Сигнал на неинвертирующий вход поступает с вывода RA0 (19) или от внутреннего источника опорного напряжения, которому соответствует несколько различных уровней. Сигнал на инвертирующий вход поступает через один из четырех аналоговых каналов: 7, 6, 5 или 1 (рис. 4.9).

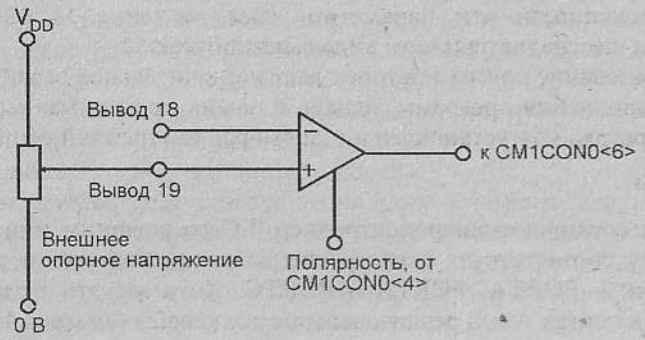



**Рис. 4.9.** Схематичное представление компаратора микроконтроллера PIC. Если разряд полярности равен 0 (неинвертированная полярность), то на выходе — сигнал высокого уровня, если входной сигнал на выводе 18 меньше опорного напряжения

Выход компаратора C1OUT может считываться как напряжение на выводе 17 или значение разряда 6 регистра CM1CON0. Для инвертировании состояния выхода используется разряд контроля полярности. Опорное напряжение устанавливается с помощью управляющего регистра VRCON (адрес 118h). Соответствующая процедура поясняется ниже.

В качестве примера настроим C1 следующим образом:

- на неинвертирующий вход сигнал подается через вывод RA0 (19);
- на инвертирующий вход сигнал подается через вывод RA1 (18);
- результат направляется в разряд 6 регистра CMCON0.

Полярность должна быть неинвертированной. При работе с микросхемой обычного аппаратного компаратора типа LM311 подключения должны соответствовать рис. 4.9.

После сброса все разряды регистра CM1CON0 устанавливаются в 0. Нам необходимы следующие установки:

- <7> = 1 — включение компаратора;
- <6> = 0 — выходной разряд;
- <5> = 0 — результат — в CM1CON0<6>, а не на вывод 17;
- <4> = 0 — для неинвертированной полярности;
- <3> = 0 — без функции;
- <2> = 1 — используется внутреннее опорное напряжение;
- <1:0> = 00 — вход — вывод 18.

Установки <1:0> для приема сигналов с других входов: 01 — вывод 15; 10 — вывод 14; 11 — вывод 7. Для того чтобы сконфигурировать компаратор 1 согласно представленному выше описанию, двоичный



байт 10000100 (шестнадцатеричное 84) должен быть сохранен по адресу 119h.

Аналогично, для компаратора 2 байт 10000101 (или 85h) сохраняется по адресу 11Ah, что соответствует приему входного сигнала с вывода 15.

Поскольку мы используем внутреннее опорное напряжение, это должно быть запрограммировано в регистре VRCON. Он может обеспечивать постоянное опорное напряжение 0,6 В или переменное в диапазоне от 0 В до 0,71875 × $V_{DD}$. Опорное напряжение подается на инвертирующий вход.

Для настройки обоих компараторов на переменное внутреннее опорное напряжение разряды в VRCON должны быть установлены следующим образом:

- <7> = 1 — использование переменного опорного напряжения для компаратора 1;
- <6> = 1 — использование переменного опорного напряжения для компаратора 2;
- <5> = 0 — выбор нижнего диапазона (1 — для верхнего диапазона);
- <4> = 0 — постоянное опорное напряжение блокировано.
- <3:0> — четыре разряда выбора диапазона переменного опорного напряжения. В нижнем диапазоне опорное напряжение равно (данная величина)/24 × $V_{DD}$, а в верхнем — $V_{DD}/4$ + (данная величина)/32 × $V_{DD}$.

Например, слово 11001100 (CCh), сохраненное по адресу 118h, настроит оба компаратора на использование переменного опорного напряжения в нижнем диапазоне. Разряды <3:0> содержат 1100 (десятичное 12), поэтому опорное напряжение равно 12/24 × $V_{DD}$, т.е. половине напряжения питания.

На выходе компаратора бит 0 или 1, который можно считать в разряде <6> регистров CM1CON0 и CM2CON0. Оба выхода можно одновременно считывать как разряды <7> и <6> третьего регистра: CM2CON1, адрес 11Bh.

Также как и в случае с регистрами, непосредственно связанными с компараторами, мы должны активизировать входы для аналоговых напряжений. Это реализуют через регистр ANSEL, адрес 11Eh.

Вывод 18 — вход AN1, а вывод 15 — вход AN5, поэтому мы устанавливаем в 1 разряды <1> и <5>. Соответствующий байт будет иметь вид 00100010 или 22h.

Все регистры, связанные с компараторами, находятся в банке 2, поэтому программа настройки компараторов 1 и 2 при опорном напряжении, равном половине напряжения питания, будет иметь вид:

```
bsf status, 6       ; Страница 2.
movlw 22h           ; Определяем AN1 (вывод 18) и AN5
movwf ansel         ;   (вывод 15) как аналоговые входы
movlw 84h           ; Настройка компаратора 1
movwf cm1con0
movlw 85h           ; Настройка компаратора 2
movwf cm2con0
movlw CCh           ; Установка опорного напряжения для обоих
movwf vrcon         ;   компараторов
bcf status, 6       ; Возвращаемся на страницу 0
```

Кстати говоря, все листинги в этой главе предполагают, что адреса регистров были объявлены с использованием директивы EQU в начале программы (см. следующую главу), или же что они определены компилятором. Если при запуске данной подпрограммы микроконтроллер уже находится в режиме страницы 1, то вначале должен быть обнулен разряд 5 регистра STATUS.

Для опроса выхода компаратора 1 используйте следующие строки:

```
bsf status, 6       ; Страница 2
movlw 40h           ; Разряд 6 данного байта равен '1'
andwf cm1con0, w    ; Считываем выходной бит, результат - в w
bcf status, 6       ; Назад на страницу 0
```

Операция логического "И" устанавливает или сбрасывает разряд нуля регистра STATUS, который затем может быть опрошен обычным способом для определения дальнейших действий.

### Аналого-цифровые преобразователи

Аналоговый входной сигнал подается на преобразователь через любой из 14 каналов. Сейчас мы будем использовать только канал AN0, которому соответствует вывод 19. При подаче питания все выходные каналы микроконтроллера PIC автоматически устанавливаются как аналоговые входы. Регистр выбора аналоговых входов ANSEL, который находится в банке 2, используется для установки каналов в качестве цифровых входов. Если разряд <0> содержит 1, то AN0 (вывод 19) — аналоговый вход. В то же время это отключает слабое подтягивающее сопротивление, а также прерывание по изменению состояния.

Следующий шаг — выбор входного канала и других функций с помощью регистра ADCON0. Установки, используемые в этой книге, имеют вид:



- <7> = 0 — результат преобразования выравнивается по левому краю (см. ниже);
- <6> = 0 — использование напряжения питания в качестве опорного;
- <5:2> = 0000 — выбор AN0 в качестве входного канала (вывод 19) (эти разряды могут принимать и другие шестнадцатеричные значения для каналов от AN0 до AN11 — 1011);
- <1> = 1 — установите этот разряд для начала преобразования, а затем считайте его; он остается в единичном состоянии, пока выполняется преобразование, а затем переходит в нулевое состояние, когда преобразование завершено (разряд GO/DONE);
- <0> = 1 — включает преобразователь (0 — выключение). В общем случае, установка <0> настраивает преобразователь на канал AN0, а установка <1> начинает преобразование.

Заключительный шаг — выбор тактовой частоты для преобразования. За это отвечают разряды <6:4> регистра ADCON1. В наших схемах не требуется высокого быстродействия, поэтому мы можем использовать невысокую тактовую частоту, т.е. разряды ADCON1<6:4> можно оставить в нулевом состоянии, как после сброса. Другими словами, в отношении тактовой частоты преобразования никаких действий не требуется.

Результат преобразования — 10-разрядный, поэтому для его сохранения требуются два регистра: ADRESH и ADRESL. При выбранном выравнивании по левому краю (см. выше), восемь старших разрядов находятся в ADRESH, а два младших — в разрядах <7:6> регистра ADRESL. Разряды <5:0> считываются в ADRESL как 000000. Вот как это выглядит:

```
   ADRESH      ADRESL
XXXXXXXX  XX000000
|10-разрядный результат|
```

Если высокая точность не требуется (или невозможна), то содержимое ADRESL можно проигнорировать. Это дает восьмиразрядную точность (т.е. 1/256 или приблизительно 0,4 %), что выше точности, которую могут обеспечить большинство датчиков.

Использование канала AN0 и чтение результата меньшей точности оказывается довольно простым делом, поскольку ничего не нужно делать с рядом регистров. Единственные регистры, которые должны быть установлены, — это ANSEL в банке 2 и ADCON0 в банке 0. Результат считывается из регистра ADRESH, который также находится в банке 0. Соответствующая простая подпрограмма:

```
        bcf status, 5
        bsf status, 6       ; Банк 2
        bsf ansel, 0        ; Выбор канала AN0
        bcf ststus, 6       ; Возврат к банку 0
        bsf adcon0, 0       ; Включение преобразователя
        bsf adcon0, 1       ; Запуск преобразования
wait:
        btssc adcon0, 1     ; Если преобразование заверешено
        goto wait           ; Если не завершено
        movf adresh, w      ; Считываем 8-разрядный результат
                            ;   в рабочий регистр
```

Результат в рабочем регистре готов к обработке.

### Передача данных с помощью USART

Микроконтроллер 690 содержит встроенный универсальный синхронно-асинхронный приемопередатчик (сокращенно — USART). Мы используем его для последовательной (т.е. побитной) передачи байта данных в USART другого микроконтроллера PIC. Второй USART получает последовательные данные побитно и преобразует их в байт, который может быть считан из регистра. Передача данных может осуществляться по одному проводу (если линия 0 В будет общей для обоих PIC) или по радиоканалу.

Выход USART — RB7 (вывод 10), а вход — RB5 (вывод 12). Доступны синхронный и асинхронный режимы работы USART, однако мы рассмотрим только асинхронный, в котором один USART ожидает неопределенный период времени получения информационного сигнала от другого USART.

Как передатчик, USART управляется регистром TXSTA, расположенным в банке 1. Настройки, которые мы используем, имеют следующий вид:

- <7> = 0 — для асинхронного режима не используется;
- <6> = 0 — восьмибитная передача;
- <5> = 1 — разрешение передачи;
- <4> = 0 — асинхронный режим;
- <3> = 0 — отключение бита символа окончания;
- <2> = 0 — низкая скорость;
- <1> — устанавливается в 1, если сдвиговый регистр передачи пуст, и в 0, если он заполнен;
- <0> = 0 — девятый бит, если он используется.

Все, что мы должны сделать для передачи байта, — поместить его в регистр данных TXREG и установить разряд <5> регистра TXSTA.



Поскольку мы используем устанавливаемый по умолчанию асинхронный режим, нет никакой необходимости настраивать регистр управления скоростью обмена (BAUDCTL).

После короткой задержки для завершения передачи (обычно только одного байта) данные считываются из регистра RCREG в другом микроконтроллере PIC. Для настройки USART в качестве приемника регистр RCSTA загружают следующим образом:

- < 7 > = 1 — активизирует последовательный порт;
- < 6 > = 0 — прием восьми бит;
- < 5 > = 0 — не используется;
- < 4 > = 1 — включает приемник;
- < 3:0 > = 0 — не используются.

Подпрограмма конфигурирования USART имеет следующий вид:

```
bsf status, 5       ; Банк 1
bsf trisb, 7        ; RB7 - вход (автоматически изменяется
                    ;   на выход)
bsf trisb, 5        ; RB5 - вход
clrf txsta          ; Настраивает, но не включает передатчик
bcf status, 5       ; Назад к банку 0
bsf rcsta, 7        ; Настраивает, но не включает приемник
```

USART готов к работе, за исключением того, что он не включен. Для его включения используются следующие строки:

```
bsf status, 5       ; Банк 1
bsf txsta, 5        ; Включение передатчика
bcf status, 5       ; Банк 0
```

Подпрограмма передачи одного байта:

```
        movlw XXh       ; XX - это байт в w
transmit
        btfss pir1, 4   ; Если регистр очищен для использования
        goto transmit
        movwf txreg     ; Для передачи байта
        return
```

При этом предполагается, что передающий микроконтроллер PIC знает, что второй PIC готов к приему. Возможно, ему предварительно придется послать байт, указывающий на готовность к приему.

Подпрограмма приема байта:

```
receive
        bcf status, z       ; Сброс флага нуля
        btfss pir1, 5       ; Новые данные приняты?
        return              ; Нет, нужно попробовать позже
        btfss rcsta, 2      ; Проверка на ошибку кадрирования
        goto noferr         ; Ошибка кадрирования отсутствует
```

```
      movf rcreg, w        ; Байт в w
      bcf status, z        ; Флаг нуля сброшен - ошибка
      goto orerr           ; Контроль флага переполнения
noferr
      movf rcreg, w        ; Байт в w
      bsf status, z        ; Установка флага нуля для
                           ;   индикации корректного байта
orerr
      btfss rcsta, 1       ; Проверка на ошибку переполнения
      return               ; Ошибка переполнения отсутствует
      bcf rcsta, 4         ; Очистка непрерывного приема
      bsf rcsta, 4         ; Разрешение (очистка флага
                           ;   переполнения)
      return
```

При выходе из этой подпрограммы z = 1, если в регистре w содержится новый корректный байт. Вызовите ее, проверьте z и используйте байт в w или вызовите подпрограмму опять, и так — до тех пор, пока не будет получен корректный байт.

Представленные подпрограммы иллюстрируют настройку USART и передачу/прием данных. Другой вопрос — как вписать их в программу для реализации обмена данными с другим микроконтроллером, не прекращая выполнение программы.

### Память данных

Микроконтроллер 16F690 предоставляет 256 байт памяти данных EEPROM, которая может хранить данные долгое время. В отличие от обычной памяти SRAM, используемой в качестве временного хранилища, данные в памяти EEPROM могут храниться не менее 40 лет или до момента перезаписи.

Самое важное — данные в этой памяти не теряются при выключении питания, поэтому она очень полезна в роботах, которые могут обучаться или же другими способами узнавать о наилучших способах реакции на сложившуюся ситуацию. Они не забывают то, что они узнали, и могут переносить приобретенные знания из одного сеанса работы в следующий.

Память данных адресуется в диапазоне от 0 до 0FFh. Считать данные из нее проще, чем записать. Подпрограмма для чтения данных из памяти EEPROM выглядит следующим образом:

```
bsf status, 6      ; Банк 2
bcf status, 5
movlw адрес_для_чтения
movwf eeadr        ; Адрес - в регистр eeadr
bsf status, 5      ; Банк 3
bcf eecon1, 7      ; Доступ к памяти данных
bcf eecon1, 0      ; Чтение данных
bcf status, 6      ; Банк 2
movf eedat, w      ; Данные в w
bcf status, 5      ; Банк 0
```



Запись данных осуществляется строгой последовательностью команд:

```
      bsf status, 6        ; Банк 2
      bcf status, 5
      movlw адрес_для_записи
      movwf eeadr          ; Адрес - в регистр eeadr
      movlw данные_для_записи
      movwf eedat          ; Данные - в регистр eedat
      bsf status, 5        ; Банк 3
      bcf eecon1, 7        ; Доступ к памяти данных
      bsf eecon1, 2        ; Разрешение чтения
      bcf intcon, 7        ; Запрет прерываний. Здесь
                           ;   начинаются требуемые коды
      movlw 055h
      movwf eecon2         ; Запись 55h
      movlw 0aah
      movfw eecon2         ; Запись aah
      bsf eecon1, 1        ; Запись данных. Требуемые коды
                           ;   здесь заканчиваются
writing
      btfsc eecon1, 1      ; Становится 0, когда данные
                           ;   полностью записаны
      goto writing         ; Пока данные записываются
      bcf eecon1, 2        ; Запрет записи
      bcf status, 5        ; Банк 0
      bcf status, 6
```

### Регистры PIR1 и PIR2

Два регистра запросов на прерывание от периферийных устройств (PIR) полезны, когда необходимо знать текущее состояние встроенных периферийных устройств микроконтроллера PIC. К ним относятся аналого-цифровые преобразователи (АЦП), приемник и передатчик USART и компараторы. При возникновении прерывания микроконтроллер PIC переходит к подпрограмме обслуживания прерывания и считывает данные из нескольких источников, чтобы выяснить причину прерывания. Однако чтение из этих устройств до того, как они завершили текущую операцию, дает неправильный результат. Мы должны позволить им завершить операцию, и в этом помогают регистры PIR.

Регистр PIR1 содержит три флага, указывающих на готовность или неготовность в данный момент времени АЦП, приемника и передатчика. Разряд <6> (ADIF — флаг прерывания от АЦП) устанавливается в 1, когда текущее преобразование завершено. Если при его считывании обнаруживается, что он содержит 0, то это означает, что опрос следует повторить позже. Разряд <5> (RCIF), устанавливается в 1, когда буфер приемника полон, а 0 указывает на то, что данные все еще накапливаются в буфере, поэтому попытку следует повторить позже. Этот флаг автоматически сбрасывается после чтения данных буфера из регистра RCREG.

Разряд <4> (TXIF) устанавливается в 1, когда буфер передатчика пуст и ожидает получения очередных данных. В противном случае этот разряд содержит 0, указывая на то, что буфер заполнен и ожидает передачи данных.

Регистр PIR2 содержит два флага, указывающих на то, что изменился сигнал на выходе компаратора. Разряд <6> (C2IF) устанавливается в 1 при изменении выходного сигнала компаратора 2, а разряд <5> (C1IF) — компаратора 1.

Эти разряды остаются в единичном состоянии до тех пор, пока не будут обнулены с помощью соответствующей команды. Это означает, что нам не обязательно опрашивать их сразу же после завершения операции сравнения. Это можно сделать в любой момент позже. Однако, если данные разряды необходимо использовать для контроля последующих операций сравнения, то их следует заранее обнулить на некотором этапе.

Все перечисленные флаги устанавливаются при возникновения событий независимо от того, разрешены прерывания или нет, поэтому их можно использовать даже тогда, когда не используются прерывания. Они сообщают о возможности безопасного считывания данных из соответствующих периферийных устройств. Однако, если далее в программе разрешаются прерывания, не забудьте предварительно сбросить все флаги.

### Регистр INTCON

Этот регистр управляет обработкой прерываний. Интерес представляют следующие разряды:

- <7> (GIE) — 1 — прерывания разрешены; 0 — прерывания запрещены. Этот разряд используется для разрешения или запрета всех прерываний одной командой.



- <6> (PEIE) — 1 — разрешение прерываний от периферийных устройств, 0 — запрет таких прерываний.
- <5> (TOIE) — 1 — разрешение прерывания по переполнению таймера TR0, 0 — запрет этого прерывания.
- <4> (INT) — 1 — разрешение прерывания по входу INT, 0 — запрет этого прерывания. Прерывание по входу INT — внешнее, возникающее при изменении состояния вывода RA2. Направление изменения, вызывающее прерывание, зависит от состояния разряда INTE (6) регистра OPTION: 1 — прерывание по нарастающему фронту, 0 — прерывание по ниспадающему фронту. По умолчанию INTE = 1.
- <3> (RABIE) — 1 — разрешает прерывание по изменению состояния порта A и порта B. Можно выбирать, какие разряды порта A активируют функцию прерывания по изменению состояния, устанавливая один либо большее число разрядов регистра IOCA.

Младшие три разряда являются флагами:

- <2> (T0IF) — устанавливается, когда переполняется таймер TM0;
- <1> (INTF) — устанавливается при возникновении прерывания по входу INT (см. выше);
- <0> (RABIF) — устанавливается при возникновения прерывания по изменению состояния порта A или B.

Как и флаги регистров PIR, эти флаги остаются в единичном состоянии до тех пор, пока не будут обнулены.

## Другие разновидности микроконтроллеров PIC

Поскольку все микроконтроллеры PIC используют один и тот же набор команд, программы, написанные для одного типа PIC, зачастую работают и на другом типе. По разным причинам может потребоваться модификация программ, однако обычно с этим связано несколько проблем.

В табл. 4.1 описаны три популярных микроконтроллера PIC, относящихся к среднему подсемейству, на которых могут выполняться по крайней мере некоторые из описанных программ для роботов без значительных изменений. Все эти типы PIC поддерживаются ассемблером MPASM, средой MPLAB, пакетом PICkit 2 и компилятором PICBASIC.

**Таблица 4.1.** Сравнение популярных микроконтроллеров PIC

| Характеристика | 16F84A | 16F628A | 16F88 |
|---|---|---|---|
| Выводы | 18 | 18 | 18 |
| Порты (разряды) | A (5), B (8) | A (8), B (8) | A (8), B (8) |

Таблица 4.1. Окончание

| Характеристика | 16F84A | 16F628A | 16F88 |
|---|---|---|---|
| ОЗУ общего назначения (байтов) | 68 | 224 | 368 |
| Память программ (слов) | 1 K | 2 K | 4 K |
| EEPROM (байтов) | 64 | 128 | 256 |
| Встроенная периферия | Один таймер | Три таймера, два компаратора, захват/сравнение, внутренний осциллятор | Два таймера, АЦП, два компаратора, захват/сравнение, внутренний осциллятор, USART |

Основная модификация затрагивает платы. Все микроконтроллеры PIC, перечисленные в табл. 4.1, имеют 18 выводов, однако разводка выводов питания в них различна (рис. 4.10 – рис. 4.12).

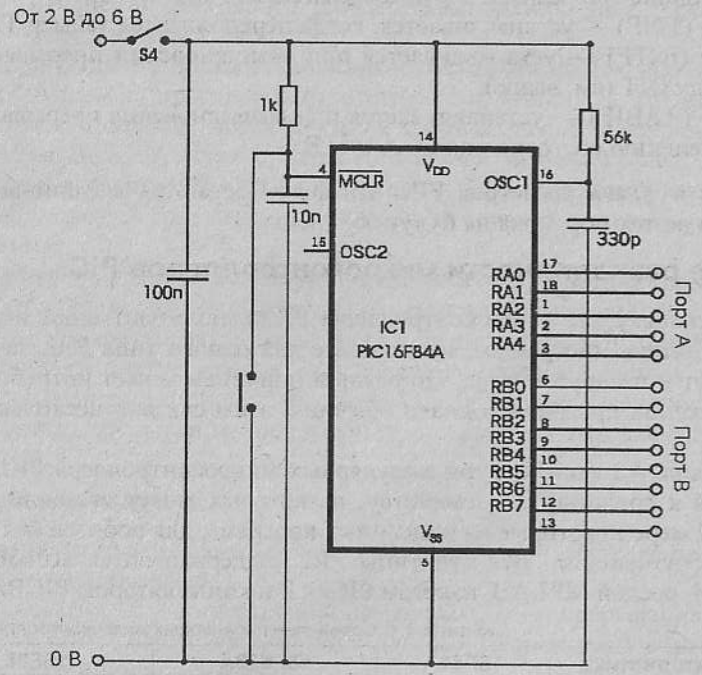



**Рис. 4.10.** Микроконтроллеру 16F84A требуется внешний осциллятор, представленный здесь RC-схемой



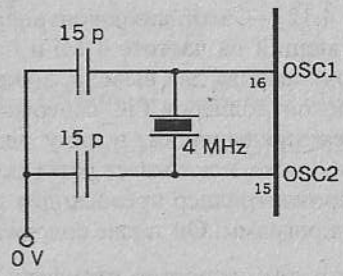



**Рис. 4.11.** Кварцевый осциллятор обеспечивает более точную синхронизацию

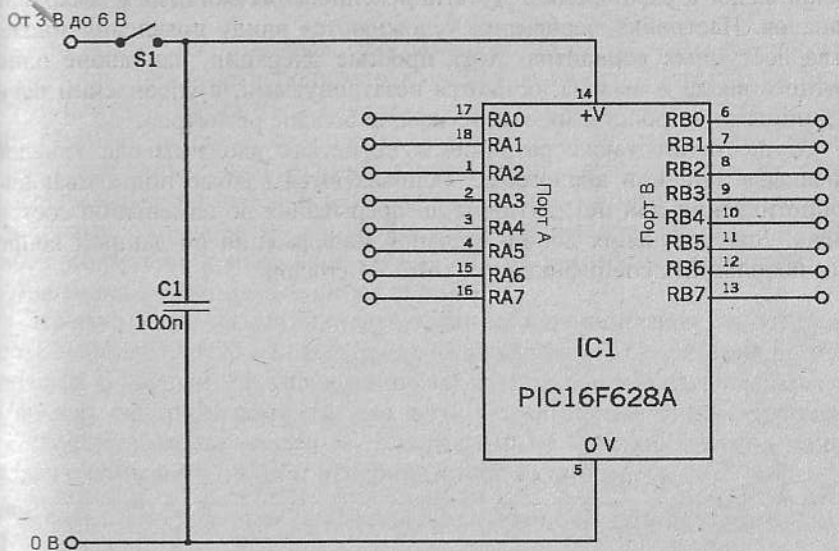



**Рис. 4.12.** У микроконтроллеров 16F628A и 16F88 совпадает разводка выводов

Рассматривая отдельные микроконтроллеры PIC, можно отметить следующие важные различия:

- **16F84A** — не имеет внутреннего осциллятора, поэтому необходимо использовать внешний генератор. На рис. 4.10 показан RC-осциллятор. Программы, рассмотренные в этой книге, предполагают, что микроконтроллер 16F690 работает от своего внутреннего тактового генератора с частотой 4 МГц. Для того чтобы рассмотренные программы работали с той же скоростью на микроконтроллере 16F84A, требуется внешний кварцевый осциллятор, работающий на частоте 4 МГц (максимальная частота для 16F84A).

- **16F628A** (см. рис. 4.12) — этот микроконтроллер имеет внутренний осциллятор, работающий на частоте 4 МГц по умолчанию. Это освобождает для ввода-вывода два вывода, доводя их общее число до 16. Этот тип микроконтроллеров PIC содержит больше периферийных устройств, чем предыдущий, в силу чего весьма популярен. Схема, приведенная ниже, показывает разводку его выводов.
- **16F88** — этот микроконтроллер превосходит другие (но не 16F690) большой памятью программ. Он также содержит АЦП.

Увеличение числа и возможностей периферийных устройств неизбежно ведет к расширению мультиплексирования входных и выходных каналов. Настройки периферии усложняются ввиду появления множества доступных вариантов. Хотя простые операции, наподобие однобитного ввода и вывода, остаются незатронутыми, в управлении периферийными устройствами задействовано больше регистров.

Существуют также различия в свойствах ввода-вывода каналов: цифровые они или аналоговые; используются слабые подтягивающие сопротивления или нет; активны ли прерывания по изменению состояния… Для получения дополнительной информации по данным вопросам используйте спецификации с сайта Microchip.

# Глава 5. Программирование микроконтроллеров PIC

## Программные сегменты

Как видно из примеров программных проектов, представленных в этой книге, некоторые подпрограммы используются многократно. Они могут частично отличиться в деталях, но по существу они — те же. Таким образом, рационально сохранить их в виде отдельных текстовых файлов для дальнейшей загрузки в текущий проект. Хотя может потребоваться незначительное редактирование, это — ничто, по сравнению с усилиями, затрачиваемыми на ввод подпрограмм каждый раз "с нуля". Такой подход не только экономит время на разработку программ, но и уменьшает вероятность ошибок ввода.

Программные сегменты, перечисленные в данной главе, — это подборка наиболее часто используемых подпрограмм. Так, листинг 5.1 начинается с "шапки", которая не является строго необходимой, однако позволяет быстро вспомнить, для чего предназначена данная программа, спустя несколько месяцев. Каждая строка "шапки" должна начинаться с символа ";", чтобы игнорироваться ассемблером.

Файл `Leader.asm` также находится на прилагаемом к книге компакт-диске в папке `Сегменты`.

**Листинг 5.1. Директивы для настройки типичной программы**

```
;*****************************************************************
;
; Имя файла: Leader.asm
;
; Директивы для настройки типичной программы.
;
;*****************************************************************

list p=16F690
__config 0x30c4

; Банк0
```

**Листинг 5.1. Продолжение**

```
  pcl         equ 02h
  status      equ 03h
  porta       equ 05h
  portb       equ 06h
  portc       equ 07h
  intcon      equ 0bh
  pir1        equ 0ch
  pir2        equ 0dh
  rcsta       equ 18h
  txreg       equ 19h
  rcreg       equ 1ah
  adresh      equ 1eh
  adcon0      equ 1fh
; Банк1
  option_reg  equ H'01'
  trisa       equ H'05'
  trisb       equ H'06'
  trisc       equ H'07'
  ioca        equ H'16'
  txsta       equ H'18'
  adres1      equ H'1e'
  adcon1      equ H'1f'
; Банк2
  eedat       equ 0ch
  eeadr       equ 0dh
  vrcon       equ 18h
  cm1con0     equ 19h
  cm2con0     equ 1ah
  cm1con1     equ 1bh
  ansel       equ 1eh
  anselh      equ 1fh
; Банк3
  eecon1      equ 0ch
; Коды пунктов назначения и код флага нуля
  w           equ 00h
  f           equ 01h
  z           equ 02h
; Метки
  delay0      equ 20h
  delay1      equ 21h

  goto start
  org 04h
  goto start

start
```



**Листинг 5.1. Окончание**

```
  goto $

; Подпрограммы

delay
  decfsz delay0, f
  goto delay
  decfsz delay1, f
  goto delay
  return

  end
```

Первая активная строка листинга сообщает ассемблеру, какой будет использоваться микроконтроллер PIC. Далее следует директива конфигурации, устанавливающая ключевые свойства работы PIC.

При использовании среды разработки, наподобие MPLAB IDE, вводить представленный далее перечень директив излишне. Все эти данные содержатся в файле компоновщика, который автоматически загружается при запуске ассемблера. Необходимо ввести только определения для меток регистров, использованных в программе. Например, метка `delay0` используется в подпрограмме задержки (см. главу 9).

При работе просто с ассемблером типа MPASM список определений необходим. Он поучается таким длинным ввиду множества задействованных регистров. Хотя маловероятно, что все они могут понадобиться в одной программе, мы сделали перечень исчерпывающим, чтобы раз и навсегда обеспечить все возможные определения. В любой программе лишние строки можно при желании просто удалить.

За определениями регистров следуют определения отдельных кодов: `w`, `f` и `z`. Хотя их включать и необязательно, это считается целесообразным. Некоторые из команд PIC имеют форму "сделай это и помести результат туда-то". Под "туда-то" подразумевается рабочий регистр (`w`) или регистр, задействованный в данный момент (`f`). Например, для декрементирования значения в регистре с именем `count` и размещения результата в рабочем регистре служит команда

```
decf count, 0
```

Эта команда оставляет `count` без изменений. Для записи результата в `count` без изменения состояния рабочего регистра служит команда

```
decf count, 1
```

Используя эквиваленты w и f вместо 0 и 1, эти же команды можно записать как:

```
decf count, w
decf count, f
```

Это значительно проясняет (для программиста), что происходит.

За определения следует список меток для регистров общего назначения, используемых в программе. Их диапазон начинается по адресу 20h и (поскольку таких регистров 96) заканчивается адресом 7fh. Первые 80 из этих адресов (от 20h до 60h) доступны в банках от 0 до 2, однако последние 16 адресов — только в банке 0.

В листинге 5.1 используются только два помеченных регистра, требуемые для подпрограммы задержки. Затем стандартным способом указывается начальный адрес, резервируя четыре строки для перехода к подпрограмме обслуживания прерывания (если таковая присутствует) в случае возникновения прерывания.

На метке start команда goto $ посылает процессор обратно в начало, формируя непрерывный цикл. Это может не сработать с некоторыми ассемблерами. В таком случае придется заменить указанный оператор на goto start. Любая версия данного оператора носит временный характер, предотвращая от входа в подпрограмму задержки. Метка start определяет позицию для начала ввода собственной программы.

Файл Leader.asm включает в себя подпрограммы задержки, которые необходимы в любой программе. Обычно подпрограммы размещают в конце листинга, хотя некоторые программисты предпочитают располагать их ближе к началу. Их положение не играет никакой роли для характера работы программы.

И еще одна существенная деталь: программа всегда должна заканчиваться директивой end.

## Входы и выходы

Программы обычно начинаются с очистки портов и установки каждого канала как входа или выхода. Необходимо также решить, какие входные каналы должны быть цифровыми, а какие — аналоговыми. Если канал представляет собой цифровой вход, то требуется ли для него слабое подтягивающее сопротивление? Типичная подпрограмма настройки показана в листинге 5.2.

Этот фрагмент можно также найти на прилагаемом к книге компакт-диске в файле Настройки.txt в папке Сегменты.



**Листинг 5.2. Типичная подпрограмма настройки**

```
start
  bcf intcon, 7      ; Запрет прерываний
  bsf status, 5      ; Банк 1
  movlw 3eh          ; <1:5> - входы, остальные - выходы,
  movwf trisa        ;   для порта A
  movlw 20h          ; <5> - вход, остальные - выходы,
  movwf trisb        ;   для порта B
  clrf trisc         ; Все выходы
  movlw 06h          ; <2:1> - прерывание по изменению
  movwf ioca         ;   состояния, порт A
  bcf status, 5      ; Банк 2
  bsf status, 6
  clrf ansel         ; Для цифрового ввода-вывода

  bcf status, 6      ; Банк 0
  clrf porta
  clrf portb
  clrf portc
```

Теперь мы осторожно настраиваем различные банки регистров специального назначения. Даже хотя файл компоновщика, возможно, снял необходимость в вводе списка определений, перечень регистров, представленный в листинге 5.1, может оказаться полезным в качестве напоминания о том, в каком банке находится каждый регистр. Так, например, регистр INTCON присутствует во всех четырех банках, чего нельзя сказать о некоторых других регистрах. При установке регистра важно указывать правильный банк.

Банк выбирается путем установки или обнуления разрядов 6 и 5 регистра STATUS. После включения питания оба эти разряда устанавливаются в 0, что выбирает банк 0 (банк по умолчанию).

Установка STATUS <5> в 1 изменяет STATUS <6:5> на 01, что выбирает банк 1. Здесь находятся регистры трех состояний, которые делают каждый канал порта входом или выходом. После подачи напряжения питания все разряды устанавливаются в 1, т.е. все каналы — входы. При обнулении некоторого разряда соответствующий канал становится выходом. Например, если все разряды порта A должны быть входами, регистр TRISA остается в состоянии по умолчанию. Далее, пусть в порте B все разряды — выходы, кроме разряда 5. Таким образом, ему соответствует маска 0010000 или 20h. Это значение помещается в рабочий регистр w командой movlw 20h, а затем переносится в регистр TRISB командой movwf trisb. Каналы порта C должны быть выходами, поэтому мы просто обнуляем все разряды командой clrf trisc.

Слабое подтягивающее сопротивление — это эквивалент высокоомного резистора, включенного между входным выводом и положительным полюсом питания. Оно обеспечивает считывание входного сигнала как единичного, если только внешние схемы не понизят его до уровня, достаточного для считывания логического нуля. Эта функция заменяет внешние подтягивающие резисторы, хотя они могут оказаться необходимыми, если входной сигнал зашумлен резкими всплесками напряжения.

Слабые подтягивающие сопротивления доступны в портах A и B, но не в порте C. В листинге 5.2 они по умолчанию отключены, поскольку установлен разряд общего разрешения 7 в настроечном регистре OPTION_REG. Если подтягивающие сопротивления необходимы, то обнулите этот разряд для разрешения *всех* слабых подтягивающих сопротивлений по умолчанию, а затем обнулите соответствующие разряды в регистрах WPUA и WPUB.

В данном примере все каналы порта A установлены как входы и подключены к ключам, которые при замыкании подключают соответствующий канал к 0 В. Это состояние по умолчанию приемлемо для порта A. В порте B все каналы являются выходами, кроме разряда 5.

В данном проекте используются два микропереключателя, которые необходимо контролировать для формирования прерываний. Они связаны с разрядами 1 и 2 порта A, следовательно, код настройки имеет вид `00000110` или 06h. Это значение помещается в регистр прерывания по изменению состояния IOCA. Настройка данного прерывания не учитывается до тех пор, пока не будет установлен разряд общего разрешения прерываний (GIE). Это можно сделать непосредственно, установив разряд GIE командой

```
bsf intcon, 7
```

В рассматриваемом примере прерывания пока не требуются, поэтому мы оставляем разряд GIE в нулевом состоянии.

Все входы в этом приложении должны быть цифровыми, однако в микроконтроллере 16F690 все каналы, которые могут работать как входы АЦП, по умолчанию аналоговые. Если мы хотим сделать их цифровыми входами общего назначения, то должны обнулить соответствующие разряды в регистрах ANSELH и ANSEL. В данном примере мы не используем АЦП. Выберите банк 2 и очистите весь регистр ANSEL, превращая все входные каналы в цифровые. Делать что-либо с регистром ANSELH нет необходимости, поскольку все каналы — выходы, кроме канала RB5, который находится под контролем USART.



На этом начальные установки завершены. Возвратившись к банку 0, микроконтроллер будет готов к выполнению главной программы, однако предварительно в качестве меры предосторожности следует очистить все выходные каналы, если только нет какой-либо особой причины не делать этого.

## Подпрограмма выбора режима

Память микроконтроллера PIC достаточно большая для того, чтобы разместить несколько различных программ (если, конечно, они не слишком объемные). По этой причине удобно было бы иметь какой-либо механизм для выбора любой из программ непосредственно во время работы микроконтроллера.

Первые строки программы после метки `start` обычно инициализируют порты и устанавливают любые опции, актуальные для всей программы. Сразу же после них следует подпрограмма выбора режима. Если в микроконтроллер записана только одна программа, данная подпрограмма не используется.

Типичная подпрограмма выбора режима показана в листинге 5.3. В данном примере два переключателя выбора режима подключены к выводам RC0 и RC1, как в проекте "Искатель" (см. главу 9).

**Листинг 5.3. Пример подпрограммы выбора режима**

```
  btfsc portc, 1    ; Проверка разряда выбора режима 1
  goto bit1hi
  btfsc portc, 0    ; Проверка разряда выбора режима 0
  goto mode2
  goto mode1

bit1hi
  btfsc portc, 0    ; Проверка разряда выбора режима 0
  goto mode4
  goto mode3
```

В программе затем происходит ветвление на четыре подпрограммы ("режима"), следующие в листинге друг за другом (каждый начинается с метки `modeX`).

На рис. 5.1 показана блок-схема этой подпрограммы. Это хороший пример использования команд ветвления. Здесь присутствуют три точки перехода в двух направлениях, выводящие программу на четыре режима. Очень важно, чтобы подпрограммы были полностью отделены друг от друга. Микроконтроллер PIC не должен переходить из одного режима в следующий по листингу, однако подпрограммы могут совместно использовать другие подпрограммы (например, временной задержки).

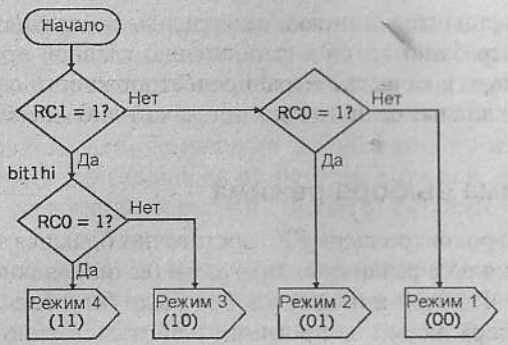



**Рис. 5.1.** Блок-схема подпрограммы выбора режима

## Команды ветвления

Набор команд PIC включает в себя две команды, одна из которых использовалась в подпрограмме выбора режима (см. листинг 5.3), а другая имеет обратное ей действие, т.е. проверяет разряд и пропускает следующую команду, если он содержит 1 (btfss). При реализации блоков принятия решений в блок-схеме используют только эти две команды.

В листинге 5.3 команда использовалась напрямую. Микроконтроллер PIC проверяет состояния разряда и выполняет соответствующее действие. Обычно с помощью такой команды контролируют разряд, который был установлен или обнулен в результате предыдущей операции. Очень часто проверяют флаг нулевого результата (разряд <2> регистра STATUS), именно поэтому мы создали особое определения для z.

На флаг нуля влияют 15 команд. Это — операции, которые могут иметь нулевой результат. Например, decf декрементирует указанный регистр и, если результат равен 00h, устанавливает флаг нулевого результата. Если результат не равен 00h, то этот флаг сбрасывается. Мы используем состояние данного флага для ветвления в одном из направлений, если z = 1 и — в другом, если z = 0. Соответствующая подпрограмма может выглядеть следующим образом:

```
decf count, f       ; Декрементирование регистра count
                    ; и запись результата в count
btfsc status, z     ; Проверка флага нуля и пропуск следующей
                    ; команды, если он сброшен
goto finish         ; Переход к метке finish, если значение
                    ; count равно 00h
```

Среди других команд, которые могут привести к изменению состояния флага нулевого результата, можно назвать сложение, вычитание



и логические операции, так что рассмотренная подпрограмма весьма полезна.

Иногда команды пропуска неудобны тем, что пропускают только одну командную строку. Действие в случае отсутствия пропуска может потребовать двух командных строки, а иногда — даже больше. В рассмотренном примере очевидно, что подпрограмма finish потребует использования нескольких командных строк, поэтому для перехода к той части программы, которая может содержать сколько угодно строк, пришлось использовать команду goto.

Иногда бывает необходимо разместить одну из двух различных констант в рабочий регистр в зависимости от состояния некоторого разряда. Например, регистр может устанавливаться в 90h, если разряд содержит 0, или в 60h, если разряд содержит 1. Проблема заключается в том, что запись константы в регистр требует использования двух командных строк, а не одной. Дело в том, что соответствующее значение вначале помещается в w (movlw), а затем переносится из w в требуемый регистр (movwf).

Обходное решение заключается в предварительной загрузке w перед считыванием разряда:

```
movlw 90h           ; Помещаем 90h в w, готовность к вращению
                    ; влево
btfsc randval, 0    ; Контроль разряда <0> регистра randval
movlw 60h           ; Изменение настройки на вращение вправо
movwf portb         ; Поворот вправо или влево
```

Этот пример взят из главы 9 (листинг режима 4). Для управления направлением поворота робота "Искатель" в порт B должно быть передано некоторое значение. Мы устанавливаем w на одну из возможных настроек (поворот влево) прежде, чем проверяем разряд случайности. Если в результате проверки окажется, что этот разряд содержит 1, то мы изменяем значение в w, чтобы повернуть вправо. При этом потребуется только одна пропускаемая командная строка. Если разряд содержит 1, то мы пропускаем команду изменения, оставляя значение для левого поворота как есть. Требуемая установка теперь находится в w, после чего она переносится в порт B, чтобы включить двигатели.

Команды опроса разряда используются в логических операциях, а также при наличии входного сигнала от кнопки или ключа. Применяют две методики использования переключателей и кнопок:

- проверка разряда по ходу выполнения программы;
- ожидание того, чтобы разряд принял конкретное значение: 0 или 1.

Подпрограмма выбора режима — это пример первой методики. Перед тем, как программа считывает состояние, переключатель уже был разомкнут или замкнут. Опрос и результирующий переход реализованы одной командой `btfsc` или `btfss`. Подпрограмма выбора режима запускается в самом начале программы. Зачем тратить переключатели на отдельную подпрограмму? Те же переключатели можно использовать позже для выбора других опций. Например, в ходе игры ключ может замыкаться для того, чтобы сказать роботу: "Я сделал мой ход — теперь твоя очередь".

Ожидание специфических входных значений подразумевает зацикливание микроконтроллера PIC. Рассмотрим простейший способ сделать это:

```
waiting btfsc portb, 4    ; Проверка сигнала от ключа на RB4
        goto waiting      ; Проверить снова: на входе 1 ?
```

Эта подпрограмма ожидает, пока вход перейдет в состояние логического нуля. Если на входе присутствует слабое подтягивающее сопротивление, то нам достаточно просто установить ключ между выводом и линией 0 В. Программа ожидает замыкания ключа. То же самое справедливо и для кнопки.

Простая подпрограмма, показанная выше, имеет свои недостатки, и главный из них — микроконтроллер PIC работает очень быстро. Если мы используем кнопку, а в программе присутствует несколько подпрограмм ожидания, то PIC, возможно, перейдет к следующей подпрограмме ожидания, в то время как мы все еще жмем на кнопку для первой подпрограммы. Прежде, чем продолжать выполнение программы, необходимо контролировать моменты нажатия *и отпускания* кнопки,.

Расширим рассматриваемую подпрограмму следующим образом:

```
waitinglow  btfsc portb, 4    ; Проверка входа от ключа на RB4
            goto waitinglow   ; Проверить снова: на входе 1 ?
waitinghigh btfss portb, 4    ; Проверить сигнал от ключа
            goto waitinghigh  ; Проверить снова: на входе 0 ?
```

Эти подпрограммы заставляют процессор ожидать в цикле. Ожидая, он не может делать ничего другого. Иногда это — именно то, что нужно, но бывает, что во время ожидания процессору желательно было бы еще чем-нибудь заняться. В таком случае потребуется подпрограмма, опрашивающая состояние кнопки через короткие интервалы.

Один из вариантов реализовать это — заставить микроконтроллер PIC проверять состояние входа от кнопки настолько часто, чтобы казалось, что нажатие кнопки имеет мгновенный эффект. Вот возможное решение:



```
flash bsf portc, 0      ; Включение светодиода на выводе RC0
      call delay        ; Включаем на 0,2 с
      bcf portc, 0      ; Выключаем светодиод
      call delay        ; Выключаем на 0,2 с
      btfsc portb, 4    ; Проверка входа от кнопки на RB4
      goto flash        ; Кнопка не нажата - еще одно мигание
```

Опрос — это простейшая из методик. Вы точно знаете, когда будет считываться вход. В этом случае известно, что цикл засвечивания светодиода заканчивается его отключением.

Альтернативная методика связана с использованием прерываний (прерывание по изменению состояния). Проблема заключается в том, что нельзя сказать с определенностью, на каком этапе цикла подсвечивания светодиода это произойдет. Необходимо запрограммировать микроконтроллер PIC таким образом, чтобы он выходил из цикла и выключал светодиод прежде, чем программа продолжит выполняться. Кроме того, следует тестировать входные каналы, чтобы выяснить, какой из них вызвал прерывание. Прерывания — великолепный инструмент в некоторых программах, однако их лучше не использовать, если аналогичного результата можно достичь с помощью простого опроса.

## Управление движением мобильных роботов

Для управления движением трехколесных роботов используют две основные методики.

- Два приводных колеса на одной оси (или, возможно, с дифференциалом), которые вращаются одним двигателем и обычно расположены сзади. Одно свободно вращающееся управляющее колесо обычно расположено спереди и управляется двигателем, ориентирующим его в требуемом направлении. Такое расположение колес напоминает детский велосипед и используется в роботе "Андроид".
- Два приводных колеса на отдельных осях — каждое с собственным двигателем. Колёса обычно устанавливают приблизительно на середине шасси, давать роботу возможность поворачиваться вокруг его центра. Третье колесо — просто ролик. Такой подход напоминает гусеничное транспортное средство (например, танк). Гусеницы программируют так же, как и колеса. Эта методика используется в роботе "Искатель".

Существует и третья методика, используемая в роботе "Скутер", который оснащен роликом, автоматически поворачивающий робота в одну сторону, когда тот движется в обратном направлении. Этот чрез-

вычайно простой метод проще всего реализовать. Он не требует программирования, однако имеет ограничения.

Двухколесное управление полагается на индивидуальное переключение двигателей в прямом или обратном направлении. В табл. 5.1 перечислены пять возможных действий робота. Они управляются цифровыми выходными сигналами: двигатель или работает или стоит. Если он работает, то может вращать вал в прямом или обратном направлении. Мы могли бы запрограммировать и аналоговое управление, при котором двигатели могут работать на разных скоростях. В результате робот может следовать по изогнутой траектории. Однако аналоговое управление усложняет программу, а цифрового в общем случае вполне достаточно.

**Таблица 5.1.** Цифровое управление

| Левый двигатель | Правый двигатель | Результат |
|---|---|---|
| Стоп | Стоп | Стоп |
| Вперед | Вперед | Вперед |
| Вперед | Назад | Поворот вправо |
| Назад | Вперед | Поворот влево |
| Назад | Назад | Назад |

Согласно используемой методике следование намеченной криволинейной траектории реализуется в виде последовательности коротких прямых участков, чередующихся с частыми, но малыми поворотами в требуемом направлении.

Например, для программирования следования по изогнутой влево кривой фактическая траектория должен состоять из ряда прямолинейных движений вперед длительностью примерно 0,2 с каждое (включаем оба двигателя и вызываем подпрограмму `delay`). Между каждым прямолинейным сегментом присутствует короткий поворот влево. Подобное разбиение траектории преследует и другую цель: оно позволяет опрашивать между сегментами датчики для проверки корректности направления движения робота.

Пример этого — режим 3 из программы робота "Искатель". Робот программируется таким образом, чтобы следовать по изогнутой черной линии. Между короткими сегментами движения вперед он опрашивает датчики, чтобы удостовериться все еще находится на линии. Если это не так, то происходит коррекция направления короткими поворотами влево или вправо.

Двигатели управляются четырьмя цифровыми выходами порта B. Для левого двигателя используются выходы A (с RB7) и B (с RB6). Они



идут на плату управления питанием, которая работает согласно описанию в предыдущей главе. Для того чтобы запустить левый двигатель в прямом направлении на выход A подается сигнал высокого уровня, а на выход B — низкого. Для обратного направления на выходе A — низкий уровень, а на выходе B — высокий. Если эти значения выразить в байтах, передаваемых в порт B, мы получаем `10000000` для прямого вращения вала и `01000000` — для реверсного (шестнадцатеричные 80h и 40h соответственно).

Для того чтобы робот двигался вперед, необходимо включить также и правый двигатель. Он управляется двумя выходами портов B и C (с RB5), а также — порта D (с RB4).

Полный байт для запуска обоих двигателей в прямом направлении должен переводить выходы RB7 и RB5 в состояние логической единицы: код `10100000` или a0 в шестнадцатеричном виде. Для запуска обоих двигателей в обратном направлении следует перевести RB7 и RB5 в состояние логического нуля, а RB6 и RB4 — в состояние логической единицы. Соответствующий код имеет вид `01010000` или 50 в шестнадцатеричной форме. Код 00h переводит все управляющие входы в состояние логического нуля, останавливая оба двигателя.

Типичные команды для запуска обоих двигателей в прямом направлении выглядят следующим образом:

```
movlw a0h    ; Код в регистр w
movwf portb  ; Оба двигателя включены
call delay   ; 0,2 с
clrf portb   ; Останов обоих двигателей
```

Это дает сегмент длительностью 0,2 с, однако мы могли бы называть вызываемую подпрограмму `longdelay` ("длинная задержка"), поместив перед ее вызовом подходящее значение в регистр `w`. Или же мы могли бы позволить роботу двигаться до тех пор, пока некоторое событие не прервет его движение.

Этот способ программирования обусловлен тем, что порт B предоставляет только четыре канала (именно поэтому четыре младших разряда управляющего кода всегда содержат `0`). В другом роботе можно было бы управлять двигателями от порта A или порта C, а другие каналы использовать в качестве выходов для других целей. Например, предположим, что двигатели управляются разрядами <7:4> порта C. Применяются те же самые коды, и передача значения a0h в порт C включает двигатели. Однако это выключит все устройства, связанные с другими четырьмя выходами. Самый простой способ решить эту проблему заключается в изменении разрядов индивидуально с помощью команд `bsf` и `bcf`.

Двигатели, переключаемые реле, программируют подобным же образом, однако используются другие коды. Как и при транзисторном переключении, каждый двигатель управляется двумя разрядами, из которых один контролирует включение/выключение, а другой — направление. Для одного двигателя управляющие коды представлены в табл. 5.2.

**Таблица 5.2.** Управление двигателями с помощью реле

| Реле "Вкл/выкл" ( 1 = вкл) | Реле "Вперед/назад" ( 1 = вперед) | Код (<1:0>) | Результат |
|---|---|---|---|
| 1 | 1 | 3h | Вперед |
| 1 | 0 | 2h | Назад |
| 1 | 0 или 1 | 0h или 1h | Стоп |

Управление с помощью двух двигателей реализуют путем удвоения цифр кода. Так, для движения вперед используют код 0fh, для движения назад — 0ah, для поворота влево — 0ch, для поворота вправо — 0eh, а для останова — 00h.

В проекте 6.5 мы используем три двигателя, переключаемых реле для перемещения рамы X, рамы Y и инструмента в различные точки рабочей области. Эти двигатели лебедок включаются по одному и наматывают или разматывают шнур лебедки. Параметры управления для одного двигателя представлены в табл. 5.3.

**Таблица 5.3.** Управление двигателем лебедки с помощью реле

| Реле вращения ( 1 = разматывание) | Реле "Вкл/выкл" ( 1 = вкл) | Код (<1:0>) | Результат |
|---|---|---|---|
| 1 или 0 | 0 | 0h или 2h | Стоп |
| 0 | 1 | 1h | Наматывание |
| 1 | 1 | 3h | Разматывание |

Коды из табл. 5.3 применяются, когда реле управляются разрядами <1:0>. В проекте "Портальный робот" направление вращения лебедки контролируется выводом RC7, а три двигателя управляются выводами RC4 (M1, лебедка Y), RC5 (M2, лебедка X) и RC6 (M3, лебедка инструмента) (см. главу 10). Управление шаговыми двигателями рассматривается в главе 3.

## Обнаружение объектов

Способность обнаружить объект, не вступая с ним в физический контакт, зачастую является ключевым элементом поведения робота. Для обнаружения источника света на расстоянии используется фотоэлемент, наподобие фоторезистора, а ультразвуковый датчик может обнаружи-



вать относительно большие твердые объекты на расстоянии вплоть до нескольких метров. В данном разделе мы рассмотрим способ обнаружения небольших объектов на расстоянии в несколько сантиметров (например, игровая фигура, которая должна быть поднята схватом робота).

Для определения наличия объекта для подъема мы используем простую стратегию. Луч светодиода направляется туда, где предположительно находится объект. Объект отражает свет от светодиода обратно к роботу, и фоторезистор воспринимает отраженный свет. В комнате могут также присутствовать и другие источники освещения, поэтому необходимо различать их излучение и свет, отраженный от объекта.

Решение заключается в попеременном включении и выключении светодиода с постоянным опросом сигнала на выходе фоторезистора (рис. 5.2).

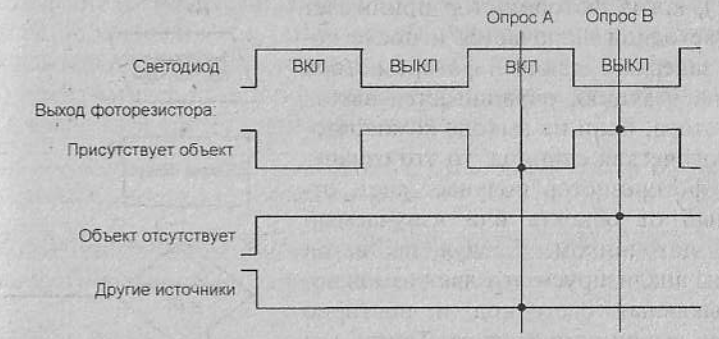



**Рис. 5.2.** Выходной сигнал компаратора для различных условий

Фоторезистор управляется выходным каналом, который поочередно устанавливается в 0 и 1 для включения и выключения светодиода. Подпрограмма мерцания должна включать короткие задержки, чтобы дать фоторезистору время на реакцию. Фоторезистор подключен к аналоговому входному каналу, который опрашивается компаратором. Вместо этого мы могли бы использовать цифровой вход от микросхемы операционного усилителя (см. рис. 3.19). В любом случае выходной сигнал компаратора падает, когда фоторезистор принимает свет, отраженный от объекта или идущий от других источников.

Если объект присутствует, то сигнал от компаратора падает, когда светодиод включен, и единичный, когда светодиод выключен. Если никакого объекта нет, то выход компаратора остается постоянно в единичном состоянии. Это также происходит, если объект слишком далеко для того, чтобы отразить достаточно света к фоторезистору. Источник

яркого света дает на выходе компаратора постоянное состояние логического нуля, которое сохраняется и при выключенном светодиоде. Эти три случая реализует подпрограмма, показанная на рис. 5.3.

Показанная блок-схема определяет простую подпрограмму, результат работы которой записывается в регистр `object`, который выполняет роль флага. Если объект обнаружен, то в этом регистре должно оказаться значение 01h.

Компаратор программируется таким образом, чтобы выдавать высокий уровень (`1`), когда фоторезистор принимает свет. Светодиод включается и после короткой задержки, дающей фоторезистору время на реакцию, опрашивается выход компаратора. Если на выходе компаратора — логическая единица, то это означает, что фоторезистор получает свет, отраженный от объекта или излучаемый другим источником. Следуя по ветви "Да", мы анализируем эти две возможности, выключая светодиод и повторно проверяя выход компаратора. Теперь мы ожидаем на выходе компаратора сигнал логического нуля. Если это так, то переменная `object` устанавливается в 01h, указывая, что объект обнаружен.

Программа далее, вероятно, должна реализовать подъем объекта, если переменная `object` равна 01h, или выполнить что-либо другое.

Если любая из проверок дает результат "Нет", то подпрограмма переходит в конец, где переменная `object` сбрасывается в 00h. Обратите внимание на то, что мы должны обнулить эту переменную явно (она может быть в состоянии 01h в результате предыдущей проверки).


**Рис. 5.3.** Обнаружение объекта. Может случиться так, что другой источник света перестанет светить на фоторезистор в момент выключения светодиода. Возможно, это маловероятно, однако для защиты от этой ошибки перед установкой или сбросом флага следует повторить две проверки. Результат "Нет" для любой из четырех проверок сбрасывает флаг



## Обход объектов

Прежде, чем объект можно обойти, его необходимо обнаружить. Эта операция, по сути, аналогична подпрограмме обнаружения объектов, однако вместо некоторого действия над объектом робот должен сдать назад и выполнить обходной маневр. Практический пример подробно рассматривается в главе 6 для робота "Скутер".

Альтернативные датчики, используемые для обхода объекта, — это бамперы, как в роботе "Искатель". Программировать их просто, поскольку все, что требуется, — это опросить одноразрядный вход и выполнить соответствующее действие.

## Музыкальные тона

Если робот оснащен динамиком, то он может воспроизводить музыку, или, по крайней мере, генерировать тоновый сигнал заданной частоты и длительности. Таким образом он может общаться с вами.

Подпрограмма `tonesound` (листинг 5.4) включает и выключая динамик на заданной частоте указанное число раз.

Эту подпрограмму можно также найти на прилагаемом к книге компакт-диске в файле `tonesound.txt` в папке `Сегменты`.

**Листинг 5.4. Подпрограмма `tonesound`**

```
; Регистры, необходимые для генерирования тона:

outloop     equ 22h
lendata     equ 23h
inloop      equ 24h
loop1       equ 25h
loop2       equ 26h
data1       equ 27h
datafinal   equ 28h
finalloop   equ 29h

; Установка регистров

  movlw 064h          ; lendata = 100 десятичное
  movwf lendata
  movlw 09h           ; data1 = 9
  movwf data1
  movlw 0ch           ; datafinal = 12 десятичное
  movwf datafinal

playit
```

**Листинг 5.4. Окончание**

```
  movlw 05h          ; Установка счетчика внешнего цикла
  movwf outloop

next1
  movf lendata, w    ; Установка счетчика внутреннего цикла
  movwf inloop
next2
  bsf portc, 0       ; Динамик вкл.
  call half
  bcf portc, 0       ; Динамик выкл.
  call half
  decfsz lendata, f ; Отсчет для внутреннего цикла
  goto next2
  decfsz outloop, f ; Отсчет для внешнего цикла
  goto next1
  return

half
  movf data1, w
  movwf loop1        ; Установка внешнего счетчика
next3
  movlw 0fh
  movwf loop2        ; Установка внутреннего счетчика
next4
  decfsz loop2, f    ; Отсчет для внутреннего цикла
  goto next4
  nop
  decfsz loop1, f    ; Отсчет для внешнего цикла
  goto next3
  nop
  movf datafinal, w ; Установка счетчика заключительного цикла
  movwf finalloop
next5
  decfsz finalloop, f  ; Отсчет для заключительного цикла
  goto next5
  nop
  return

  end
```

Перед вызовом этой подпрограммы в регистрах должны быть сохранены следующие три значения:

- `data1` и `datafinal` устанавливают частоту тона. `Data1` — это переменная внешнего цикла для двух вложенных циклов. Переменная внутреннего цикла устанавливается на 0Fh. За этими циклами следует заключительный цикл, который служит для точной подстройки частоты. В качестве переменной в нем используется `datafinal`.
- `lendata` устанавливает длительность тона. Это — переменная внутреннего из двух вложенных циклов, используемых в подпрограмме `half`. Счетчик внешнего цикла устанавливается на 05h.



Приняв, что частота осциллятора PIC составляет 4 МГц, вычисляем половину периода в микросекундах:

$$\texttt{half} = 500\,000/\text{частота}.$$

Методом проб и ошибок находим пару значений для `data1` и `datafinal`, для которых:

$$(50 \times \texttt{data1}) + (3 \times \texttt{datafinal}) + 15 = \texttt{half}.$$

Для тона подходящей длительности вычисляем:

$$\texttt{lendata} = \text{частота} \times 0{,}1.$$

Изменяя значение переменной `lendata`, можно варьировать длительность тона, однако представленное выше уравнение дает хорошую отправную точку.

В качестве примера рассмотрим вычисления для тона частотой 1 кГц:

$$\texttt{half} = 500\,000 / 1\,000 = 500 \text{ мкс}.$$

Несколько попыток дают:

$$(50 \times 9) + (3 \times 12) + 15 = 501.$$

Это близко к желаемому значению, поэтому принимаем `data1` = 9, а `datafinal` = 12. Подходящее значение для `lendata` равно 1 000 × × 0,1 = 100. В листинге 5.4 эти данные используются для формирования тона с частотой 1 кГц.

Работа программы начинается с установки вывода 0 порта C в качестве выхода. Схема для управления динамиком показана на рис. 3.35.

Листинг 5.4 может быть оформлен как подпрограмма более сложной подпрограммы, которая помещает набор значений в регистры данных и вызывает подпрограмму `playit`. Таким образом, можно заставить робота воспроизводить мелодии. В табл. 5.4 перечислены значения для получения некоторых музыкальных нот. Программируя последовательности таких тонов, вы сможете изобрести тоновый "язык", который робот будет использовать для выражения своих "чувств". Более длинная последовательность тонов становится песней.

**Таблица 5.4.** Переменные значения для музыкальных тонов, которые должны быть загружены в три регистра перед вызовом подпрограммы `playit`

| Нота | `data1` | `datafinal` | `lendata` |
|---|---|---|---|
| C | 024h | 020h | 02Ah |
| C' | 010h | 02Fh | 054h |
| E' | 0Ah | 051h | 06Ah |
| G' | 0Ah | 029h | 07Eh |
| C" | 08h | 015h | 0A8h |
| G" | 06h | 01h | 0F9h |

На рис. 5.4 показаны блок-схемы двух подпрограмм. Они показывают различные циклы, используемые при получении тона заданной частоты и длительности. Для того чтобы воспроизвести последовательно из более, чем, скажем, четырех тонов, самое лучшее — это сохранить значения в форме таблицы соответствия (см. следующий раздел). Практический пример подпрограммы, воспроизводящей тоновые сигналы, дан в описании робота "Андроид" (глава 7).

## Справочные таблицы

Когда подпрограмма должна обращаться к данным систематическим образом, то самый лучший подход — разместить данные в справочной таблице. Таблица — это блок данных в подпрограмме или блоке памяти EEPROM. Она может содержать набор числовых значений (например, переменные, используемые в подпрограмме `playit`) или кодов для различных конфигураций фигур на доске, или же закодированные результаты попыток прохождения лабиринта. К данным, хранимым в справочной таблице, осуществляется быстрый доступ.

Здесь мы говорим о том, как поместить данные в подпрограмму. Для получения доступа к данным мы объявляем переменную, которая сообщает микроконтроллеру PIC, какой конкретно элемент данных должен быть считан. Поскольку эта переменная указывает на некоторый элемент, она называется указателем. Указатель сохраняется в регистре общего назначения и инициализируется нулевым значением:

```
pointer equ 30h ; Подходит любой неиспользуемый адрес
clrf pointer    ; Обнуление
```

Подпрограмму лучше всего размещать в начале листинга (особенно, если в ней много данных). Это устраняет риск, что таблица распространится на два блока памяти программ. Первая строка имеет вид:

```
addwf pcl, w
```



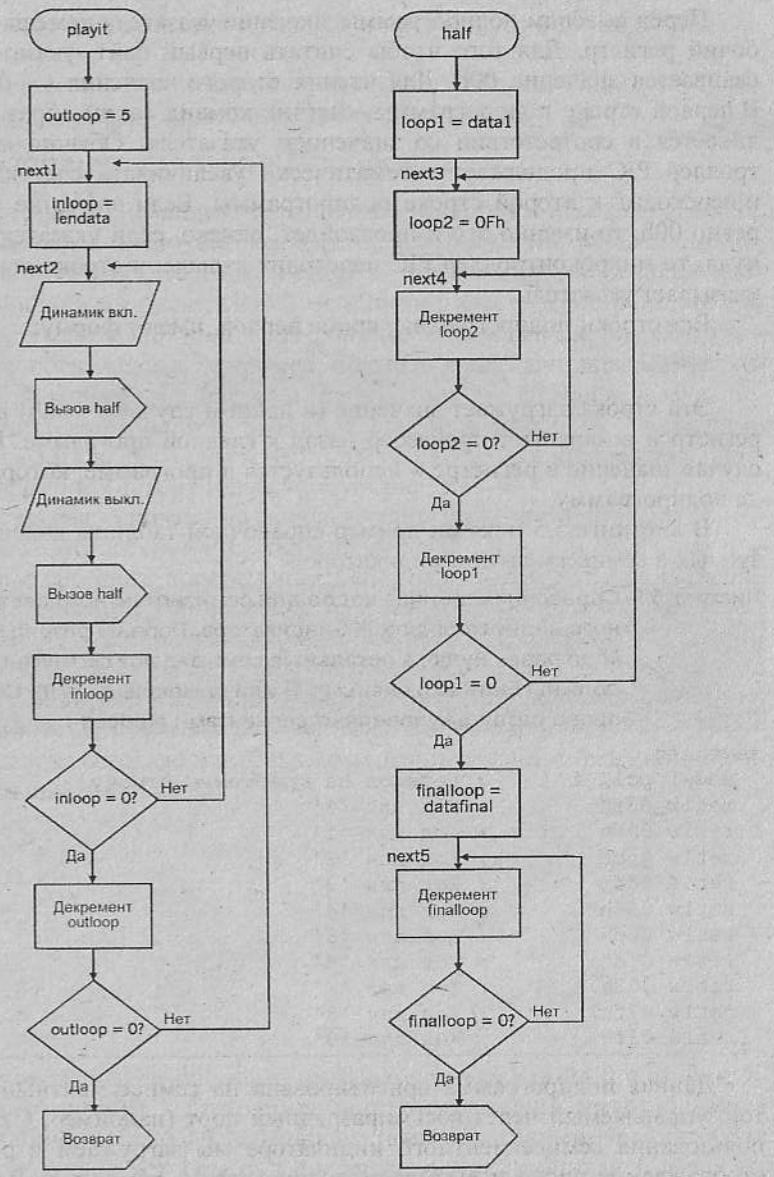



**Рис. 5.4.** Слева — подпрограмма генерирования частоты `playit`; справа — подпрограмма временной задержки на полупериод `half`

Перед вызовом подпрограммы значение указателя помещается в рабочий регистр. Для того чтобы считать первый байт, указателю присваивается значение 00h. Для чтения второго значения — 01h и т.д. В первой строке подпрограммы, счетчик команд таким образом увеличивается в соответствии со значением указателя. Обычно микроконтроллер PIC продолжает автоматически увеличивать счетчик команд и переходит к второй строке подпрограммы. Если значение указателя равно 00h, то именно это и произойдет, однако, если указатель больше нуля, то микроконтроллер PIC переходит дальше: к строке, на которую указывает указатель.

Все строки подпрограммы, кроме первой, имеют форму:

```
retlw, 0ah
```

Эта строка загружает значение (в данном случае — 0ah) в рабочий регистр и возвращает процессор назад к главной программе. В данном случае значение в регистре w используется в программе, которая вызвала подпрограмму.

В листинге 5.5 показан пример справочной таблицы кодов, используемых в семисегментном индикаторе.

**Листинг 5.5. Справочная таблица кодов для семисегментного светодиодного индикатора или ЖК-индикатора. Первый разряд каждого кода равен нулю, а остальные семь задают сегменты, которые должны быть включены (= 1) или выключены (= 0). Слева направо разряды управляют сегментами gfedcba**

```
sevenseg
  addwf pcl, f      ; Переход на требуемую строку
  retlw 03fh        ; Код для '0'
  retlw 006h        ; Код для '1'
  retlw 05bh        ; Код для '2'
  retlw 04fh        ; Код для '3'
  retlw 066h        ; Код для '4'
  retlw 06dh        ; Код для '5'
  retlw 07dh        ; Код для '6'
  retlw 003h        ; Код для '7'
  retlw 07fh        ; Код для '8'
  retlw 06fh        ; Код для '9'
```

Данная подпрограмма ориентирована на семисегментный индикатор, управляемый через восьмиразрядный порт (например, C). Для использования семисегментного индикатора мы загружаем в регистр w отображаемое число и вызываем подпрограмму sevenseg. После возврата из подпрограммы соответствующий код окажется в регистре w.



Этот код затем загружается в порт C, и число отображается на индикаторе.

Практический пример использования справочной таблицы — программа воспроизведения мелодии робота "Андроид".

## Использование двух процессоров

В системах, основанных на распределенной обработке информации, два или большее число микроконтроллеров совместно решают вычислительные задачи. Одно из преимуществ такой организации заключается в повышении вычислительной мощности и быстродействия. Это может также увеличивает количество входов и выходов системы, что позволяет ей обслуживать, например, большее число двигателей и датчиков. Программирование системы такого типа — сложная задача, поскольку микроконтроллеры работают одновременно и их необходимо синхронизовать друг с другом через набор сигналов квитирования. Достаточно простую систему иллюстрирует рис. 5.5.

Эта система основана на использовании двух физически независимых микроконтроллеров PIC, связанных проводами или по радиоканалу, которые работают попеременно. Один из них ожидает или выполняет мелкие задачи, в то время как другой занят выполнением более сложной задачи. По ее завершении, ранее занятый PIC сигнализирует ожидающему PIC, который переходит в состояние занятости, пока другой "отдыхает". Системы подобного типа легче программировать, поскольку в любой момент времени активен только один процессор.

Ниже показаны две подпрограммы для передачи и ожидания приема сигнальных импульсов:

```
pulseout
  bsf portc, 0
  call delay
  bcf portc, 0
  return

pulsein
  btfss portc, 1
  goto pulsein
  return
```

Микроконтроллер посылает импульс через канал RC0, а принимает — через канал RC1. Этот стиль программирования легко поддается планированию, однако он основан на таком программировании обоих контроллеров, при котором они придерживаются предсказуемой последовательности действий.

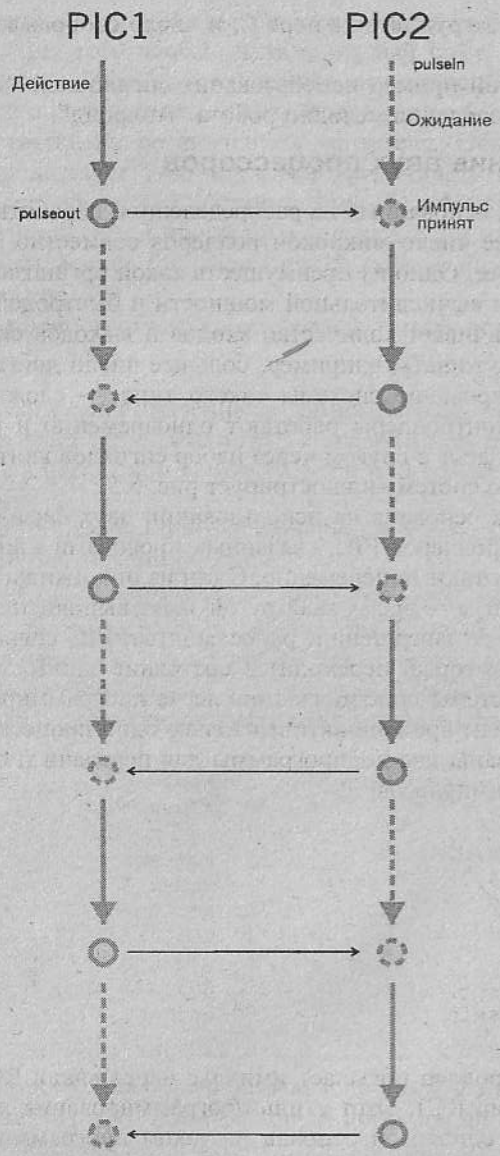



**Рис. 5.5.** Программирование двух микроконтроллеров PIC на попеременную работу. PIC посылает импульс другому, когда завершает текущую задачу, после чего ожидает (пунктирные линии) приема импульса перед возобновлением работы



Один импульс ограничен по своему смыслу. Чаще всего он означает "Моя работа завершена, теперь твоя очередь". Передача кодированного байта дает принимающему микроконтроллеру гораздо больше информации. Один байт может принимать любые значения из диапазона 0 до 255, т.е. может иметь до 256 различных смыслов. Например, если робот "Андроид" получил дополнительный двигатель для перемещения левого манипулятора, то управление этим двигателем может быть передано ведомому микроконтроллеру PIC, а остальная часть обработки информации возложена на ведущий. Ведущий PIC посылает ведомому сигналы, сообщая, какой двигатель следует включить, в каком направлении должен вращаться его вал и, возможно, — с какой скоростью. Ведущий PIC посылает один из предварительно заданных наборов кодовых байтов, а ведомый выполняет требуемые действия.

Для последовательной передачи байтов от ведущего микроконтроллера PIC к ведомому используется приемопередатчик USART. Канал RB7 (вывод 10) ведущего PIC соединен с каналом RB5 (вывод 12) ведомого. Обычное проводное соединение вполне приемлемо при условии, что оно не слишком длинное, а два микроконтроллера запитаны от одного и того же источника питания. Можно посоветовать в этом случае использовать легкий экранированный кабель, если расстояние превосходит несколько сантиметров. На *одном* из концов экран кабеля следует подсоединить к шине 0 В.

Листинг программы для ведущего микроконтроллера PIC включает в себя сегменты конфигурирования USART для его активизации и передачи байта данных. Байт должен находиться в рабочем регистре перед вызовом подпрограммы `transmit`.

Листинг программы для ведомого микроконтроллера PIC включает в себя сегменты конфигурирования USART для его активизации и приема байта данных. Если был принят новый байт, то он находится в регистре RCREG, а флаг нулевого результата регистра STATUS установлен.

## Математические операции

Система команд микроконтроллеров PIC содержит две важные операции: сложение и вычитание. Они могут быть использованы для выполнения других операций, таких как умножение и деление. Люди обычно работают в десятичной системе счисления, поэтому используемые ими алгоритмы умножения и деления основаны именно на этой системе. Логические же системы основаны на двоичном счислении, поэтому естественно, что в компьютерах операции умножения и деления — также двоичные.

Умножение на два — это просто сдвиг двоичного числа влево на один разряд. Например, двоичное 00010011 соответствует десятичному 19. Если это двоичное число сдвинуть на один разряд влево, то оно превратится в 00100110. Младший разряд в этом случае заполняется нулем. Полученное значение теперь соответствует десятичному 38, т.е. удвоенным исходным числом.

Сдвиг разрядов влево выполняется командой `rlf`. Младший разряд заполняется разрядом переноса (STATUS<0>). При использовании команды `rlf` для умножения разряд переноса следует предварительно обнулить.

Старший разряд сдвигается в разряд переноса. В рассмотренном выше примере это был 0, так что при циклическом сдвиге ничего не было потеряно. Если мы знаем заранее, что этот разряд никогда не может быть в состоянии 1, то умножение на два — это просто вопрос обнуления разряда переноса и однократного циклического сдвига влево. Если же старший разряд содержит 1, то это должно учитываться путем перемещения его в регистр, содержащий старший байт двухбайтной переменной.

Двухбайтная переменная может сохраняться в двух регистрах: `valh` и `vall`. Начните с обнуления этих двух регистров, а также — разряда переноса, после чего поместите умножаемое значение в переменную `vall`.

```
  clrf valh          ; Обнуление регистров и разряда переноса
  clrf vall
double
  bcf status, 0      ; Обнуления разряда переноса
  rlf vall, f        ; Умножение на 2, с переносом
  rlf valh, f
  bcf status, 0      ; Обнуление разряда переноса для
                     ; следующей операции
  return
```

Умножение на другое значение просто реализовать, если множитель представляет собой степень двух. Например, для того, чтобы умножить число на восемь, необходимо выполнить рассмотренный выше алгоритм три раза. Умножение на значение, которое не является степенью двух, происходит поэтапно.

В листинге 5.6 представлена подпрограмма для умножения однобайтного значения `vall` на значение второго байта (множитель). В этом примере мы умножаем 31h на 12h, чтобы получить 372h.

Эту подпрограмму можно также найти на прилагаемом к книге компакт-диске в файле `Умножение.txt` в папке `Сегменты`.



**Листинг 5.6. Программа перемножения двух однобайтных значений для получения двухбайтного результата**

```
start
  movlw 031h         ; Пример
  movwf vall
  movlw 12h
  movwf multiplier   ; Пример
  movlw 08h          ; 8 бит в байт
  movwf bitcount

multiply
  bcf status, 0
  clrf valh
  clrf templ
  clrf temph
times
  rrf multiplier, f ; Берем по порядку разряды множителя
  btfss status, 0   ; Если перенос = 1
  goto iszero       ; Если перенос = 0
  movf vall, w
  addwf templ, f    ; Добавляем значение (младший байт) в temp
  btfsc status, 0
  incf temph, f
  movf valh, w
  addwf temph, f
iszero
  call double
  decfsz bitcount, f
  goto times

finish goto finish

double
  bcf status, 0
  rlf vall, f       ; Умножение с переносом
  rlf valh, f
  bcf status, 0     ; Обнуление разряда переноса для
                    ; следующей операции
  return

end
```

Разряды множителя извлекаются по одному путем циклического сдвига вправо. Если разряд содержит 0, то ничего не предпринимается за исключением циклического сдвига влево переменной `vall` чтобы подготовить ее к следующему этапу. Если же разряд содержит 1, то зна-

чение vall добавляется к значению templ (переменная, в которой постепенно накапливается произведение).

На каждом этапе данного процесса любое переполнение vall циклически сдвигается в разряд переноса, а оттуда — сдвигается в valh, в силу чего результат представляет собой целое значение двойной точности (максимум 65 535 в десятичной форме). Никаких предупредительных мер против переполнения с переносом в третий байт не предпринимается, однако двухбайтный максимум для большинства применений этой программы достаточно велик.

Умножение может также быть выполнено путем повторных сложений исходного значения, однако метод, описанный выше, работает быстрее.

Деление на два и на степень двух легко реализуется путем циклического сдвига вправо с последующим обнулением разряда переноса. Деление на другие значения проще всего выполнить повторными вычитаниями. Делитель вычитается из делимого до тех пор, пока не будет получен отрицательный результат. Количество вычитаний подсчитывается, — это и будет частное (делимое/делитель). Подпрограмма, показанная в листинге 5.7, реализует однобайтное деление.

Эту подпрограмму можно также найти на прилагаемом к книге компакт-диске в файле Деление.txt в папке Сегменты.

**Листинг 5.7. Подпрограмма деления однобайтного значения на другое однобайтное значение с получением однобайтного результата**

```
start
  movlw 84h          ; Пример
  movwf divid        ; Делимое
  movlw 018h         ; Пример делителя в w
  clrf count
  call divide

finish goto finish

divide
  subwf divid, f     ; Вычитаем делитель из делимого
  btfss status, 0
  goto negative      ; Если результат отрицателен
  incf count, f      ; Количество вычитаний
  goto divide

negative
  addwf divid, f     ; Восстанавливаем делимое до последнего
                     ; положительного значения
```



**Листинг 5.7. Окончание**

```
  bcf status, 0
  rlf divid, f       ; Удваеваем это значение
  bcf status, 0
  subwf divid, f     ; Вычитаем делитель
  btfss status, 0
  return             ; Если округляется в сторону уменьшения
  incf count, f
  return             ; Если округляется в сторону увеличения

end
```

Подпрограмма возвращает результат в переменной count. Результат округляется в сторону уменьшения или увеличения до ближайшего целого числа.

Операторы "больше, чем" (>) и "меньше, чем" (<) легко программируются с помощью команды subwf. Значение a помещается в помеченный регистр (назовем его vala), а значение b находится в рабочем регистре. Результат получается путем считывания флага переноса STATUS <0>, а также — флага нулевого результата STATUS <2>.

Доступны три варианта (табл. 5.5).

**Таблица 5.5.** Использование команды subwf для сравнения

| | Разряд переноса C | Разряд нулевого результата Z |
|---|---|---|
| a > b | 1 | 0 |
| a = b | 1 | 1 |
| a < b | 0 | 0 |

Для комбинированного оператора >= ("больше чем или равно") необходимо считывать только разряд переноса. Аналогичным образом, если значение a помещено в w, и используется команда sublw, то можно сравнивать a с фиксированным значением

## Случайные числа

На первый взгляд может показаться, что случайным числам нет места в программировании роботов, однако в некоторых ситуациях они все же необходимы. Так, в игровых программах случайные числа могут имитировать подбрасывание кости или другое случайное событие. В тех программах, в которых роботы учатся на своем опыте, первичная схема поведения часто носит случайный характер, однако робот учится модифицировать ее для получения более эффективной модели поведения.

Другое использование случайных чисел заключается в том, чтобы сделать поведение роботов более "человеческим". Люди часто ведут се-

бя *как бы* случайно, но так только внешнее ощущение, поскольку мы просто не знаем, что у них на уме. Умеренное применение случайных чисел наделяет таким же качеством и роботов.

Хотя для удобства мы используем термин "случайные", на самом деле числа, генерируемые подпрограммой, абсолютно предсказуемы. Просто последовательности чисел (нулей и единиц) настолько длинные до повторения, что выглядят случайными. Правильным термином будет "псевдослучайные" числа.

Подпрограмма имитирует аппаратный генератор (псевдо)случайных чисел, в котором используется сдвиговый регистр (рис. 5.6).


**Рис. 5.6.** Генератор случайных чисел может быть построен на базе микросхемы сдвигового регистра и логического элемента "Исключающее ИЛИ"

Принцип заключается в том, что содержимое двух регистров (m и n), составляющих сдвиговый регистр, подвергается операции логического "ИЛИ", и результат помещается обратно в первый регистр (0). Это формирует псевдослучайную последовательность чисел. Длина этой последовательности зависит от того, какие регистры участвуют в операции. Как показано на рис. 5.6, а также — в листинге 5.8, при m = 5 и n = 6, последовательность будет повторяться через каждые 127 битов.

Эту подпрограмму можно также найти на прилагаемом к книге компакт-диске в файле `Random.txt` в папке `Сегменты`.

**Листинг 5.7. Подпрограмма формирования псевдослучайных чисел**

```
  movlw 0DBh          ; Исходное значение (выбрано произвольно)
  movwf random

monte
  clrf bitn           ; Обнуление регистров
  clrf bitm
  bcf status, 0       ; Флаг переноса = 0
  btfss random, 5     ; n = 1?
```



**Листинг 5.7. Окончание**

```
  goto findm          ; Нет: теперь находим m
  bsf bitn, 0         ; Да: делаем bitn = 1

findm
  btfss random, 6     ; m = 1?
  goto xorem          ; Нет: переход на 'логический элемент'
  bsf bitm, 0         ; Да: делаем bitm = 1

xorem
  movf bitn, w
  xorwf bitm, w       ; Исключающее ИЛИ bitn и bitm
  addlw 0ffh          ; Устанавливаем флаг переноса, если w = 1
  rlf random, f
```

Генератор имеет произвольный набор битов в своих регистрах, с которого он начинает, когда переменной `random` (эквивалентна сдвиговому регистру) присваивается исходное значение. Это значение имеет в своем составе по крайней мере одну единицу. Если начальное значение равно 00000000, то схемы логического "ИЛИ" не дадут на своем выходе ничего, кроме нулей.

Следующий шаг заключается в присвоении переменным `bitm` и `bitn` значений, находящихся в регистрах 5 и 6. Они подвергаются операции логического "ИЛИ" командой `xor` (эквивалент логической схемы "Исключающее ИЛИ"). Результат этой операции (0 или 1) сохраняется в `w`, а затем складывается с константой ffh. Если результат выполнения операции исключающего "ИЛИ" равен 0, то значением `w` остается ffh, а флаг переноса остается сброшенным. Если же он будет равен 1, то значение изменяется на 00h, и флаг переноса устанавливается. Команда циклического сдвига влево, которая записывает флаг переноса в регистр 0, формирует новый набор значений в переменной `random`.

Для считывания случайного числа используйте команду `btfss` или `btfsc` для получения значения одного из разрядов переменной `random`. Затем выполните действия, зависящие от того, будет ли полученный бит равен 0 или 1. Биты также могут выбираться группами по два или больше для получения случайных чисел от 0 до 3 (отбор по два бита), от 0 до 7 (три бита) и т.д.

Листинг генератора случайных чисел всегда начинается с одного и того же начального значения. Это формирует одну и ту же последовательность всякий раз при запуске генератора. Лучше использовать начальное значение, которое получается случайно. Один из способов обеспечить это — начать с известного начального значения, однако за-

тем обеспечить работу генератора в цикле в ожидании нажатия кнопки для останова генератора и продолжения работы программы. В точности момент нажатия кнопки неизвестен, поэтому генерирование случайных чисел будет начинаться с неизвестного начального значения.

## Калибровка системы

Значения некоторых переменных, используемых в наших программах, корректны только для наших версий роботов. Проблема — в управлении двигателем. Когда приводной двигатель включается на конкретный отрезок времени (скажем, 0,2 с), то расстояние, пройденное роботом, будет зависеть от типа двигателя, передаточного числа коробки передач, напряжения питания, диаметра колес и шин. Ваш робот, наверняка, будет отличаться от собранного нами (по крайней мере, в одном из перечисленных аспектов). То же относится и к другим переменным (например, к выходным данным аналоговых датчиков).

Обычно мы можем решить, каким образом откорректировать значения переменных, наблюдая робота в действии. Робота, движущегося по прямой, можно запрограммировать на движение в течении 10 путем вызова подпрограммы `longdelay`, вызывающей 50 раз подпрограмму `delay`. Рабочий регистр при этом перед вызовом подпрограммы `longdelay` загружается значением 32h. Если, например, робот перемещается вперед на расстояние, вдвое превышающее ожидаемое, то измените программы, загрузив `w` половинным значением, т.е числом 19h.

Иногда нам необходимо опрашивать значение переменной в то время, когда робот в действии. Например, может считываться результат аналого-цифрового преобразования с последующим его сохранением в памяти EEPROM. В конце прогона программы поместите микроконтроллер PIC в программатор и считайте сохраненное значение. Затем соответствующим образом откорректируйте программу.

## Программное обеспечение взамен аппаратного

Обычно можно использовать более простые датчики или же уменьшить их количество, если мы компенсируем это более изощренным программированием и более сложным поведением робота. В качестве примера можно привести поведение робота "Искатель", основанное на отслеживании линии. Оно основано на использовании всего двух датчиков.

Мы могли бы сэкономить пространство, входные каналы и стоимость, если используем всего один датчик. При использовании только *одного* фотоэлемента робота необходимо запрограммировать так, чтобы



он держал датчик прямо над траекторией. Если датчик определит, что робот отклонился от траектории, то робот начнет рыскать из стороны в сторону, пытаясь вновь найти траекторию и остаться на ней. Программируйте робота таким образом, чтобы он запомнил, какое направление смещения успешно вернуло его на траекторию. Затем, когда робот в следующий раз отклонится от траектории, он прежде всего попробует то направление, которое в прошлый раз привело его к успеху. Если он все еще находится на той же криволинейной траектории (что вполне вероятно), то такое поведение поможет ему быстрее вернуться на траекторию.

Это всего лишь один пример взаимозависимости механического, электронного и программного аспектов робототехники. Хотя соответствующие вопросы и анализируются в отдельных главах данной книги, эффективная робототехническая конструкция должна учитывать все три составляющие в их единстве.

# Часть III

# Проекты

***В этой части***

- Робот “Скутер”
- Робот “Андроид”
- Робот-игрушка
- Робот “Искатель”
- Портальный робот

# Глава 6. Робот “Скутер”

Этот робот имеет недорогую конструкцию, а его электронные схемы предельно просты. Модульный электронный дизайн удешевляет проект еще и возможностью замены датчиков и перепрограммирования микроконтроллера PIC, в результате чего получается несколько различных роботов по цене, незначительно превышающей одного.

Ниже перечислены основные свойства робота “Скутер”, однако никто не мешает добавлять и другие функции после сборки ключевых конструкций:

- корпус из готовой пластмассовой коробки;
- питание от батареи на 4,5 В или 4,8 В;
- три колеса: два задних приводных, один смещенный ролик впереди для простого управления;
- микроконтроллер PIC 16F690A;
- датчики: переключатель, адаптивные фотоэлементы;
- исполнительные механизмы: двигатель, светодиоды, зуммер/сирена.

Программы:

- сообщение “Hello, World!” (“Привет, мир!”);
- калибровка компаратора;
- поиск источника света;
- фотоэлемент как датчик приближения;
- обход препятствий;
- использование АЦП.

## Механика

Сборка робота из готовой коробки ускоряет процесс. Это также удобно для тех, кто не слишком опытен (или не слишком заинтересован) в сборке “с нуля”. Опытный образец робота был собран в пластмассовом пищевом контейнере с защелкивающейся крышкой, купленном в местном универсаме. Контейнер был квадратным с размером стороны около 120 мм и в 50 мм в высоту. Благодаря зеленой прозрачной пластмассе и оранжевой крышке, робот, бесцельно рыская по комнате, мигая светодиодами, выглядел довольно эксцентрично.

Магазины для моделистов предлагают множество пластмассовых коробок различных форм и размеров. Выберите квадратную, возможно немного большую, чем та, которую мы использовали для опытного образца.

При выборе коробки, избегайте тех, которые сделаны из ломкой пластмассы, наподобие ударопрочного полистирола. Пищевые коробки обычно делают из довольно жесткой, и при этом — слегка гибкой пластмассы, которая легко сверлится и режется. Именно этот тип пластмассы подходит для нашего проекта. Впрочем, иногда из-за тепловыделения при сверлении отверстия получаются с рваным краям, однако в целом аккуратно обработать пластмассу — не проблема.

Некоторые производители предлагают коробки с сеточной разметкой на крышке, что удобно при сверлении отверстий. Они также могут содержать выемки для монтажа печатных плат и встроенный батарейный отсек.

### Колеса и управление движением

Робот "Скутер" — трехколесный, что придает ему устойчивости на немного неровных поверхностях. Интересно противопоставить приводную систему "Скутера" с совершенно другой системой робота "Искатель". Два передних колеса "Скутера" установлены на одном валу, поэтому для приведения их в движение требуется только один двигатель и одна схема управления им. Это упрощает и удешевляет робота.

Третье колесо — это ролик. Его конструкция зависит от используемого типа малых колес. Мы использовали пару колес из конструктора Lego® с шинами диаметром 25 мм. Колеса крепятся на двух выступах пластмассового кубика из конструктора со стороной 16 мм. Второй кубик такого же типа крепится над первым, что дает возможность закрепить болтами колесную сборку на пластине из ПВХ (или медной полоске). Другой конец этой пластины крепится на болте, выступающем из крышки (рис. 6.1).

Робот управляется чрезвычайно простым механизмом. При движении вперед рычаг ролика выставляется в положение, показанное на рис. 6.1. Робот "Скутер" всегда движется прямо вперед. Для того чтобы повернуться, он должен немного отъехать назад, что заставляет рычаг ролика повернуться в угловую позицию (рис. 6.2). В результате робот движется назад и поворачивается одновременно.

Если он сдаст назад несколько раз, то в результате опишет небольшую окружность диаметром около 400 мм. Обычно поворот осуществляется только на небольшой угол прежде, после чего робот опять движется вперед, но уже в другом направлении.



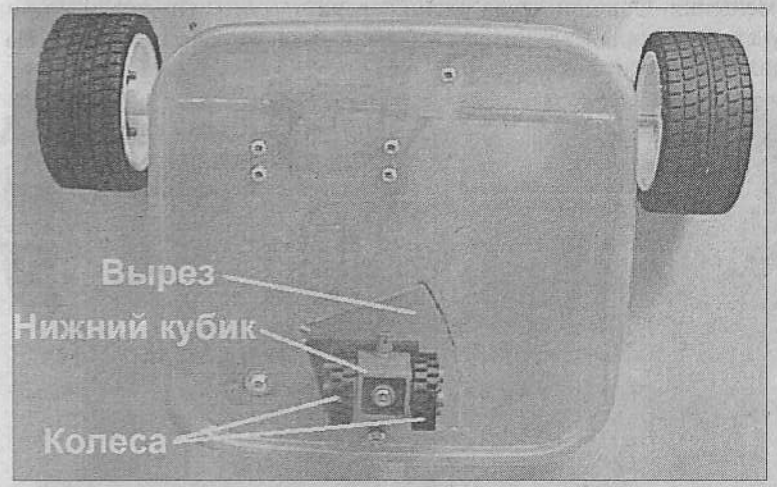



**Рис. 6.1.** Колесный узел робота "Скутер" прикреплен к крышке коробки (здесь — вид снизу). Робот движется на трех колесах, однако задний ролик, — это, фактически два колеса. Ролик вращается вокруг оси, расположенной с одной стороны продольной центральной линии. Здесь ролик показан в позиции, соответствующей движению робота прямо вперед

Данную методику нельзя отнести к настоящему управлению движением робота, но зато ее легко реализовать, она действенна и проста в программировании. Вместо нее можно использовать управление с помощью двух двигателей. В таком случае обращайтесь к описанию робота "Искатель" (глава 9). Его колесная и моторная системы могут легко быть установлены в пластмассовой коробке.

### Сборка колесной системы

Мы выбрали типичную пищевую коробку с защелкивающейся крышкой. Ее углы закруглены (см. рис. 6.1). Самый лучший способ использовать этот тип коробки — перевернуть ее. Крышка (теперь внизу) несет двигатель, приводные колеса и ролик. Печатные платы и детали робота размещаются в коробке.

Электрические двигатели вращаются слишком быстро для того, чтобы управлять колесами напрямую, поэтому необходимо встроить в робот коробку передач. Мы выбрали двигатель постоянного тока на 6 В (номинальное напряжение), укомплектованный пластмассовыми передаточными шестеренками. Они собираются так, чтобы дать два пере-

даточных понижающих коэффициента: 1:60 и 1:288. Мы собрали передачу 1:60, поскольку "Скутер" предназначен для достаточно быстрого перемещения. Ось двигателя расположена под прямым углом к приводному валу, который крепится с обеих сторон коробки передач. На его концах фиксируются колёса (рис. 6.2).

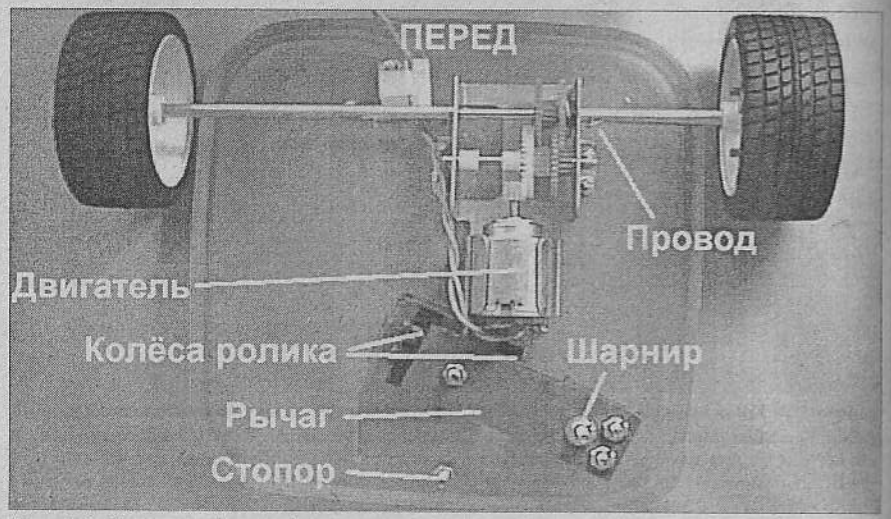



**Рис. 6.2.** Вид сверху (внутри) колесного узла, прикрепленного к крышке. Колесики ролика установлены под крышкой и двигаются в веерообразном вырезе. Рычаг, на котором установлены колеса, вращается на шарнире, представляющем собой длинный болт, зафиксированный в крышке. На этой фотографии рычаг находится в позиции, соответствующей движению робота назад. Стопор ограничивает движение рычага, когда он находится в позиции "вперед". В стопоре для позиции "назад" необходимости нет, поскольку рычагу препятствует корпус двигателя

Постоянная проблема для создателей роботов заключается в том, что вал выбранного двигателя и ступицы колес — разного диаметра. При этом крайне важно, чтобы колеса сидели на валу плотно, без проскальзывания. Это именно тот случай, когда необходима импровизация.

Мы выбрали пару колес "спортивного" вида от компании Tamiya диаметром 56 мм. Набор включает ступицы для установки колес на валу, но не включает вал. Для создания устойчивой базы приводные колеса должны быть разнесены приблизительно на 170 мм, однако выходной вал коробки передач недостаточно длинный. Для наращивания вала мы использовали с обеих сторон алюминиевые трубки диаметром 4 мм. Они с одной стороны вставлялись в ступицы, а с другой надевались на выходной вал. Отрезок трубки должен быть достаточно длинным для



того, чтобы удерживать колеса на приемлемом расстоянии от боковой стенки коробки.

Трубка диаметром 4 мм плотно садится на выходной вал и не нуждается в какой-либо поддержке. Сборка не скользит при вращении приводного вала, однако силы трения обычно постепенно ослабляют крепление. Для того чтобы предотвратить это, закрепите трубку на валу и просверлите сквозь них отверстие диаметром 1 мм (или меньшее). Вставьте в него короткий отрезок монтажного провода и загните его концы, чтобы он не выпал (рис. 6.3).

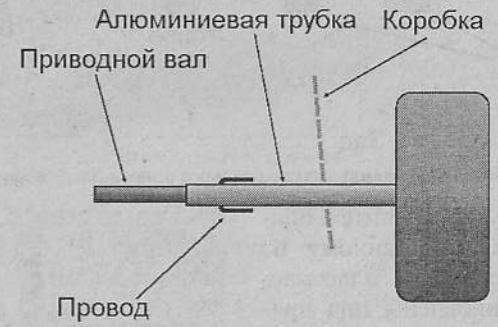



**Рис. 6.3.** Соединение одного из приводных колес с выходным валом коробки передач. Отрезок провода, пропущенный через отверстие, просверленное в приводном валу и трубке, предотвращает проскальзывание. (Не в масштабе.)

Рассмотрим альтернативу такой сборке. Как было отмечено выше, данный робот может двигаться прямо вперед или назад, поворачивая вправо. Пара основных колес может быть смонтирована под углом, в результате чего робот при движении вперед будет постоянно уходить влево. Для того чтобы придерживаться прямой, ему придется постоянно корректировать это смещение уходом на короткое расстояние назад, а затем вновь двигаться вперед. Для ухода вправо перед движением вперед необходимо проехать большее расстояние. Фактически, это обеспечивает управление типа "влево-вправо", что существенно важно, если робот будет пытаться двигаться вдоль стены.

В боковых стенках коробки прорезаются пазы, которые немного шире, чем диаметр трубки.

Ролик можно собрать несколькими способами, однако главная его особенность заключается в том, что колесо (колеса) монтируются на рычаге, вращающемся на оси, закрепленной справа или слева от продольной оси робота. Ось колеса (колес) параллельна рычагу (рис. 6.4).

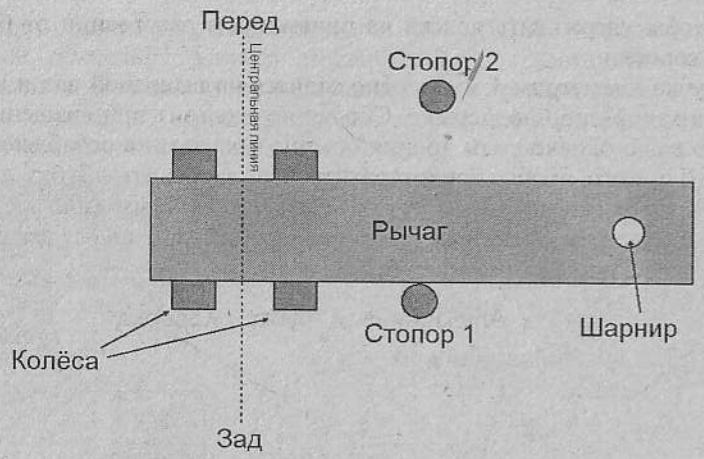



**Рис. 6.4.** Принцип эксцентрикового ролика (вид сверху)

Колесный узел крепится болтами к рычагу, вырезанному из листового металла или пластмассы. Колеса установлены под прямым углом к продольной оси рычага. На другом конце рычаг вращается на болте (рис. 6.5). Длина болта и позиция рычага на нем откорректированы так, чтобы крышка, опираясь на приводные колеса и ролик, была в горизонтальном положении.

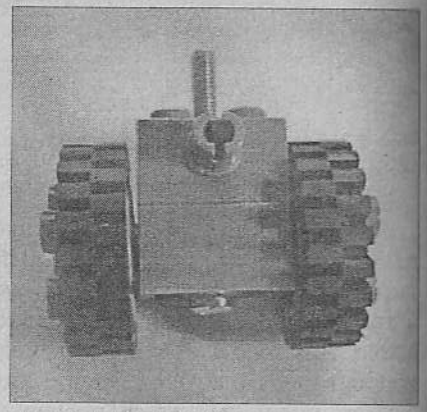


**Рис. 6.5.** Колесный блок сделан из двух идентичных кубиков из конструктора, на которых крепятся колеса. Колеса вращаются на двух креплениях на боковых стенках нижнего кубика. Верхний кубик также имеет два крепления, ориентированных перпендикулярно к креплениям нижнего. Через центральные отверстия обоих кубиков пропущен длинный болт М3, предназначенный для крепежа узла к рычагу

Рычаг может оседать под весом робота по двум причинам. Одна из них — слишком гибкая пластмасса крышки. В этом случае укрепите крышку, прикрепив болтами к крышке в точке прохождения оси маленький квадрат более жесткой пластмассы. Опытный образец робота "Скутер" не нуждался в таком укреплении.

Поскольку рычаг должен поворачиваться на оси свободно, его нельзя крепить жестко, поэтому он



может проседать. Как следствие, рычаг необходимо сделать толще. Решение показано на рис. 6.6 и рис. 6.7.


**Рис. 6.6.** Перед сверлением отверстия для шарнира два маленьких пластмассовых квадрата крепятся болтами к рычагу (при этом используются два коротких болта). Затем через все три слоя просверливается отверстие под ось

На этом мы завершаем изучение механических аспектов данного робота, не считая монтажа печатных плат, когда они будут готовы, а также некоторых других деталей, наподобие светодиодов. Платы привинчиваются болтами к крышке и ко дну коробки болтами М3 на 15 мм и соответствующими гайками.

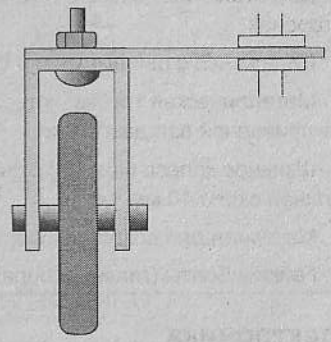


**Рис. 6.7.** Альтернативная конструкция ролика с одним колесом. Монтаж может быть выполнен с помощью алюминиевой или медной полосы шириной около 10 мм

### Резюме

Подытожим план относительно механики робота "Скутер".

- Выберите коробку подходящей (квадратной) формы, размеров (квадрат со стороной около 120–150 мм), изготовленную из подходящего материала (неломкая, поддающаяся сверлению и резке пластмасса).
- Спланируйте размещение двигателя, коробки передач, приводных колес и ролика. Попытайтесь обеспечить роботу настолько большой дорожный просвет, насколько это окажется практичным. Для устойчивости приводные колеса должны быть максимально смещены назад и разнесены.

- Передаточный коэффициент коробки передач должен обеспечивать роботу "Скутер" скорость в диапазоне от 20 до 50 см/с.
- Удостоверьтесь, что приводные колеса не скользят и не спадают с оси при вращении.
- Вырежьте отверстие в крышке для колес ролика, и установите рычаг на оси так, чтобы он не проседал.
- Установите стопоры.
- Установите узел, состоящий из двигателя, коробки передач и приводных колес.
- Вырежьте пазы в боковых стенках коробки, через которые будет проходить приводной вал.
- В заключение, проверьте, что робот ровно стоит на своих трех колесах, а колеса не трутся о коробку.

| *Список приобретений для механической части робота* |
|---|
| Коробка квадратная из нехрупкого пластика (возможно, прозрачная). |
| Двигатель постоянного тока и коробка передач с выходным валом на обеих сторонах. |
| Пара колес с шинами около 60 мм в диаметре. |
| Металлическая трубка, которая вставляется в ступицы колес и надевается на приводной вал двигателя. |
| Шкивное колесо (колеса) около 25 мм в диаметре для ролика, с валом, длиной около 40 мм. |
| Материал для создания опоры роликовых колес. |
| Гайки и болты (главным образом М3). Длина болтов — 6 и 10 мм. |

## Электроника

Робот управляется микроконтроллером PIC16F690. При небольших изменениях в программах можно воспользоваться и некоторыми другими типами PIC. Система питается от четырех металлогидридных литиевых элементов AAA или AA. Они дают выходное напряжение 4,8 В (до 5,2 В, когда только что заряженные), которое лежит в диапазоне допустимого напряжения питания для микроконтроллеров PIC (до 5,5 В).

Система реализована на трех печатных платах: контроллера (для PIC), управления двигателем и коммутационной. Используются также несколько внешних компонентов: батареи, светодиоды и фоторезисторы.



Для того чтобы сэкономить пространство и сделать робот компактным и проворным, мы использовали одножильный монтажный провод с ПВХ изоляцией. Оголенные концы проводов вставлялись в отверстия в полосах плат. Провод, который продают как монтажный или телефонный, иногда немного толстоват для того, чтобы войти в такие отверстия. Учтите это при покупке провода.

### Плата контроллера

Придерживаясь идеи маленьких компактных модулей, плата контроллера была спроектирована в минимальном варианте. По сути, это — гнездо под микросхему с 20 выводами с перфорированными полосками для организации соединений (рис. 6.8).

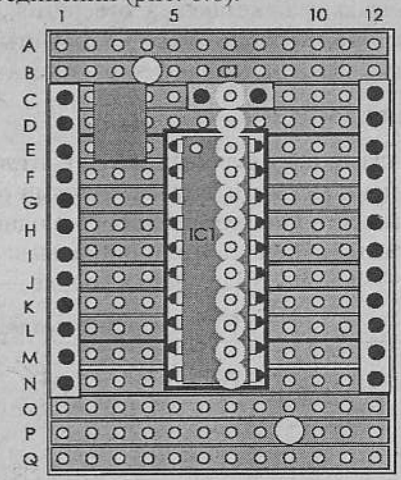



**Рис. 6.8.** Контроллер с обеих сторон содержит перфорированные полоски для организации соединений. Обратите внимание на вырезы в медных дорожках, изолирующие выводы на противоположных сторонах. Гнезда C1, D1 и E1 объединены в группу путем нанесения на соответствующие полоски на тыльной стороне платы капли расплавленного припоя. Эта группа служит для подключения положительного полюса напряжения питания и помечена кусочком красной изоляционной ленты. Полюс 0 В питания подключается к гнездам C12, D12 и E12, которые также соединены между собой каплями припоя на тыльной стороне платы. Конденсатор C1 — это миниатюрный конденсатор с полиэфирным диэлектриком на 100 нФ

### Плата управления двигателем

Эта плата (рис. 6.9) содержит Н-образный мост для управления направлением вращения вала двигателя. Она монтируется на крышке возле узла, состоящего из двигателя и коробки передач.

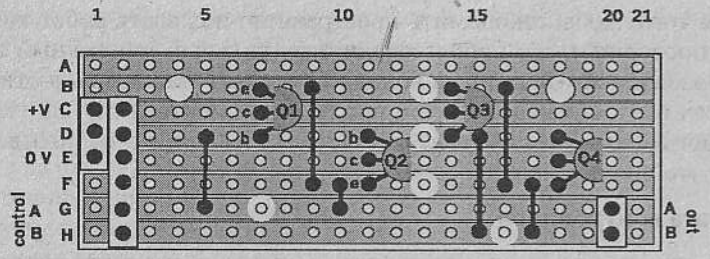



**Рис. 6.9.** Плата управления двигателем содержит два соединительных разъема для линий питания. Это позволяет ей использовать один шлейф с другими платами и модулями

Для подключения к этой плате используется тот же метод, что и в случае с платой контроллера: вставка в отверстия оголенных концов проводов. Альтернативно, провода можно припаять прямо к выводам или отдельным гнездам. Детали подключения описаны далее.

### Коммутационная плата

Так называется плата, связывающая микроконтроллер PIC с датчиками и исполнительными механизмами, отличными от двигателя. Внутри робота достаточно места для размещения еще одной платы этого типа на случай расширения системы. Принципиальная схема показана на рис. 6.10, монтажная — на рис. 6.11, плата в сборке — на рис. 6.12.

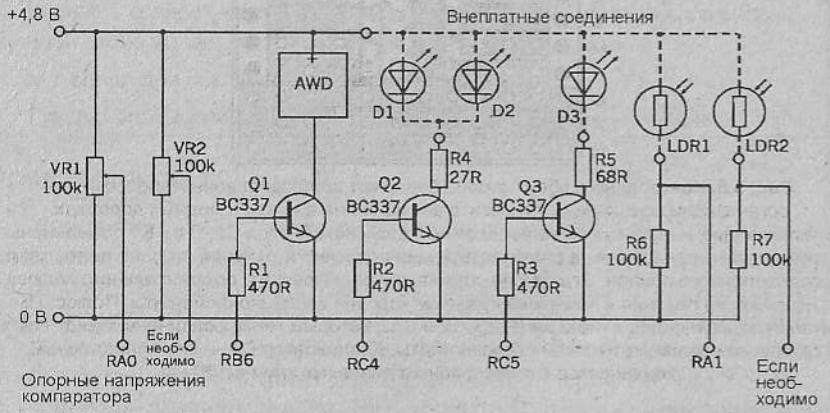



**Рис. 6.10.** Как и транзисторные ключи для светодиодов и зуммера (AWD), коммутационная плата содержит резисторы двух цепей фотоэлементов. Она также содержит два необязательных делителя напряжения для использования со схемами компаратора

Ток для светодиодов можно обеспечить непосредственно с выхода микроконтроллера PIC, однако в этом случае его сила не должна превышать 20 мА. Данный проект использует яркие светодиоды для обеспечения освещения, необходимого светочувствительным датчикам приближения. Эти светодиоды требуют до 70 мА каждый, поэтому их необходимо переключать с помощью транзисторных ключей. В опытном образце светодиоды D1 и D2 были белыми и выдавали 18 кд. Боковой светодиод D3 был синим и выдавал 10 кд.



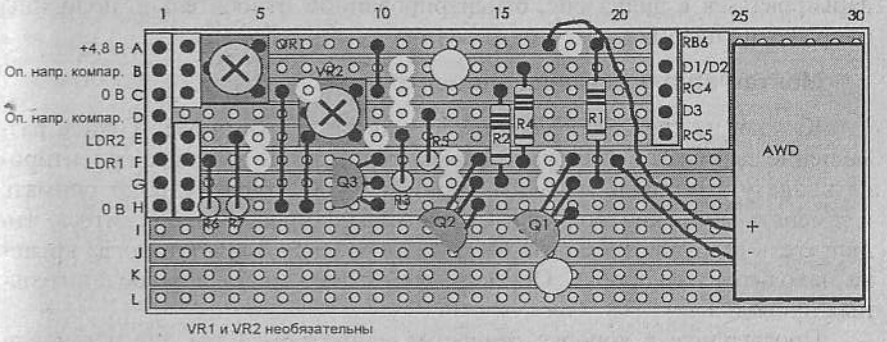



**Рис. 6.11.** Общий вид коммутационной платы. Два подстроечных резистора VR1 и VR2 необязательны

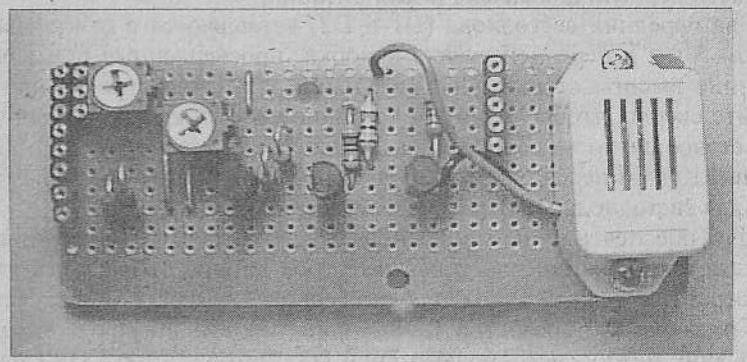


**Рис. 6.12.** Готовая коммутационная плата

Зуммер — полумеханический с потреблением тока 40 мА, а значит, для него также необходим транзисторный ключ.

Транзисторы BC337 (см. рис. 6.10) рассчитаны на ток 800 мА, что намного больше, чем рекомендуемые токи светодиодов. Вместо них подойдут почти любые n-p-n-транзисторы (например, BC548), если только не требуется включить параллельно несколько светодиодов с пиковым током 100 мА и более. Распределение выводов для BC548 совпадает

с распределением для BC337, так что для обоих типов транзисторов разводка платы одинакова.

Номиналы резисторов R6 и R7 зависят от сопротивлений фоторезисторов LDR1 и LDR2 и должны быть примерно равны сопротивлению фоторезистора в условиях освещенности, при которых, как предполагается, будет работать робот. Выходной сигнал датчика при этом будет варьироваться в диапазоне, отцентрированном относительно половины напряжения питания.

### Монтаж плат и внешних элементов

Каждая из трех плат монтируется двумя болтами M3. Точное размещение зависит от размера и формы коробки. Не забудьте ориентировать плату контроллера таким образом, чтобы было удобно снимать и заменять микроконтроллер PIC при отладке программ. Убедитесь, что двигатель и другие элементы на крышке не касаются плат, когда крышка находится на коробке. Оставьте достаточно места для соединительных проводов.

Просверлите в коробке отверстия диаметром 3 мм для установки болтов, удерживающих платы. Еще одно отверстие просверлите в нижней части коробки для выключателя питания.

Два передних светодиода (D1 и D2) вставляются в отверстия диаметром 5 мм в передней стенке коробки, просверленные примерно на половине высоты стенки. При сверлении этих отверстий немного наклоните сверло, чтобы "лучи" от светодиодов сходились, формируя одно световое пятно на объектах, расположенных приблизительно на расстоянии 100 мм перед роботом. Просверлите одно отверстие диаметром 5 мм для светодиода D3 с левой стороны робота.

Для обеспечения направленной чувствительности фоторезисторы вставляются в отрезки пластиковых трубок длиной 10 мм подходящего диаметра. Для каждой такой трубки сверлится отверстие соответствующего размера. Фоторезистор LDR1 ориентирован вперед и находится посередине между светодиодами D1 и D2. Фоторезистор LDR2 направлен влево и расположен возле светодиода D3.

### Внеплатные соединения

Платы связаны между собой одножильными монтажными проводами в ПВХ изоляции. Изоляция удаляется с концов каждого провода примерно на 5 мм. Необходимые соединения перечислены в табл. 6.1. Нарезайте провода настолько коротко, насколько это возможно.



**Таблица 6.1.** Необходимые соединения между платами робота "Скутер"

| От | | Функция | К | |
|---|---|---|---|---|
| **Плата** | **Гнездо** | | **Плата** | **Гнездо** |
| Контроллер | E1 | Положительная шина питания | Коммутационная | A1 |
| Коммутационная | A2 | | Управление двигателем | C2 |
| Управление двигателем | C1 | | Выключатель питания S1* | |
| Выключатель питания S1 (общий)* | | | Положительная клемма батареи | |
| Контроллер | E12 | Линия 0 В | Управление двигателем | E2 |
| Управление двигателем | E1 | | Коммутационная | C1 |
| Коммутационная | C2 | | Отрицательная клемма батареи | |
| Контроллер | L1 | Управление двигателем A | Управление двигателем | G2 |
| Контроллер | M1 | Управление двигателем B | Управление двигателем | H2 |
| Управление двигателем | G20 | Выход двигателя A | Клемма двигателя M1* | |
| Управление двигателем | H20 | Выход двигателя B | Клемма двигателя M1* | |
| Контроллер | G12 | LDR1 | Коммутационная | F1 |
| | F12 | Опорное напряжение компаратора 1 | | B1 |
| | N12 | Зуммер | | A22 |
| | J1 | D1 и D2 | | C22 |
| | I1 | D3 | | E22 |

* Провода припаиваются к клеммам.

Положительная шина питания также идет на внеплатные светодиоды и фоторезисторы. Для аккуратности монтажа эту линию, а также провода, идущие от компонентов, лучше всего собрать в один жгут.

Положительная линия изображена на рис. 6.13 как сплошная линия. Это провод без изоляции, идущий от анода D1 (передний правый) через переднюю часть корпуса до середины его левой стороны. Затем он продолжается до положительной клеммы выключателя питания S1. Не-

большая петля на этом уровне охватывает каждый из электронных компонентов, через которые проходит данная линия: D1, LDR1, D2, D3 и LDR2. Между этими точками на провод надет отрезок изолирующей трубки. Выводы электронных компонентов коротко обрезаны, загнуты в форме крючков и вставлены в петли провода положительной линии. Петли и крючки обжаты плоскогубцами, а места соединений пропаяны.

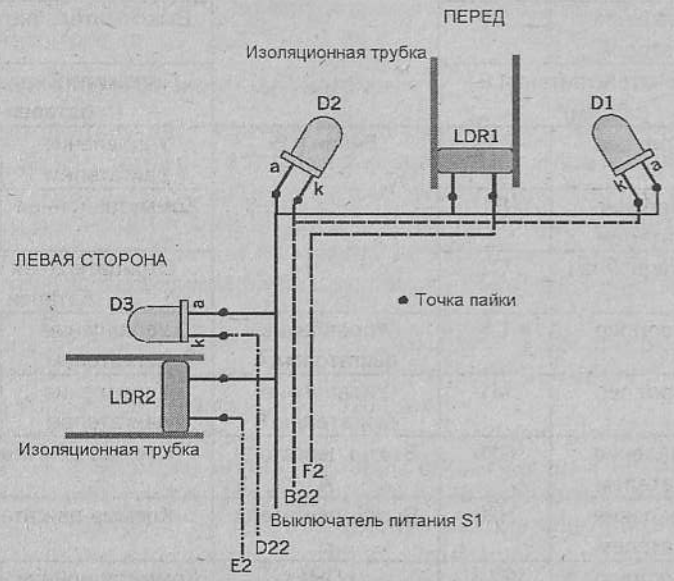



**Рис. 6.13.** Соединения между коммутационной платой и светодиодами/фоторезисторами (схематически)

Остальные четыре линии идут обратно от неположительного вывода каждого компонента к гнезду на коммутационной плате. Одна из них обслуживает два компонента: катоды D1 и D2. Эти соединения используют изолированный одножильный монтажный провод, который спирально обернут вокруг положительной линии, чтобы придать жгуту некоторую жесткость.

Кабельный жгут удерживается двумя небольшими зажимами, расположенными с обеих сторон фоторезистора LDR1. Эти самоклеющиеся зажимы не крепятся к пластмассе коробки, поэтому прикручиваются болтами M2.

Батарейный отсек удерживается отрезком ленты Velcro (рис. 6.14). В отсеке находятся четыре перезаряжаемых NiMH-элемента AAA, подключенных к схеме с помощью стандартного соединителя PP3.




**Рис. 6.14.** Содержимое коробки по окончании монтажа (вид снизу). Большинство соединений — между платой процессора и коммутационной платой, однако провода отведены от процессорного гнезда, чтобы не мешать установке и извлечению микроконтроллера PIC. Жгут к светодиодам/фоторезисторам идут влево (на фотографии — вправо) вблизи от отрезка ленты Velcro. Выключатель питания S1 можно просматривается между отрезком ленты Velcro и зуммером

На этом монтаж соединений завершен, однако на этом этапе крайне важно все проверить. Процедура тестирования выглядит следующим образом.

1. Проверьте непрерывность линии 0 В и положительной линии питания.
2. Проверьте отсутствие короткого замыкания между этими двумя линиями.
3. Без микроконтроллера PIC в гнезде подайте питание и проверьте, подается ли напряжение на каждый наплатный и внеплатный компонент.
4. В гнезде микроконтроллера PIC приложите положительное напряжение к каждой выходной линии, используя табл. 6.2, чтобы определить контрольные точки. Светодиоды и зуммер должны срабатывать. Когда линии положительного и нулевого напряжения подключаются к выводам 8 и 9, двигатель должен вращаться вперед или назад.

5. С помощью тестера удостоверьтесь, что напряжение, снимаемое с фоторезисторов, имеет приемлемое значение. При этом фоторезисторы следует осветить, а затем затенить.

### Выводы микроконтроллера PIC

В табл. 6.2 перечислены подключения к выводам микроконтроллера PIC. Ксерокопия этой таблицы, прикрепленная к стенду, будет полезной при сборке и тестировании модулей системы.

**Таблица 6.2.** Подключения к выводам микроконтроллера PIC

| Порт | Вывод | Номер | Тип | Подключение | 0= | 1= |
|---|---|---|---|---|---|---|
| A | RA0 | 19 | Аналог. вход | Опорное напряжение (AN0) | | |
| | RA1 | 18 | Аналог. вход | Передний фотоэлемент (AN1) | | |
| | RA2 | 17 | Вход | Выбор программы | Прог. 1 | Прог. 2 |
| | RA3 | 4 | Вход | | | |
| | RA4 | 3 | Вход/выход | | | |
| | RA5 | 2 | Вход/выход | | | |
| B | RB4 | 13 | Вход/выход | | | |
| | RB5 | 12 | Вход | Зарезервировано под вход USART | | |
| | RB6 | 11 | Выход | Зуммер | Выкл. | Звук |
| | RB7 | 10 | Выход | Зарезервировано под выход USART | | |
| C | RC0 | 16 | Вход/выход | | | |
| | RC1 | 15 | Вход/выход | | | |
| | RC2 | 14 | Вход/выход | | | |
| | RC3 | 7 | Вход/выход | | | |
| | RC4 | 6 | Выход | Фары | Выкл. | Вкл. |
| | RC5 | 5 | Выход | Боковые фонари | Выкл. | Вкл. |
| | RC6 | 8 | Выход | Двигатель A | Вперед | Назад |
| | RC7 | 9 | Выход | Двигатель B | Назад | Вперед |



Строки табл. 6.2 сгруппированы по портам ввода-вывода. В ней перечислены все каналы, присутствующие в PIC16F690. Вывод от RC0 до RC3 могут быть использованы как аналоговые входы компараторов или АЦП, если потребуется использовать больше датчиков. Кроме того, в табл. 6.2 показаны установки линий двигателя A и B. Для остановки двигателя обе линии должны быть переведены в нулевое или единичное состояние.

***Список приобретений для электронной части робота***

Плата контроллера:
- конденсатор с диэлектриком из полиэфира C1, 100 нФ;
- микроконтроллер PIC16F690;
- гнездо микросхемы на 20 выводов с двухрядным размещением;
- полоса 2 x 12 гнезд;
- плата с полосками: 17 полос x 12 отверстий.

Плата управления двигателем:
- Q1, Q3 — n-p-n-транзисторы BC639 (2 шт.);
- Q2, Q4 — p-n-p-транзисторы BC640 (2 шт.);
- полоса на 2, 3, и 6 гнезд;
- плата с полосками: 8 полос x 21 отверстий.

Коммутационная плата:
- R1–R3 — резисторы на 470 Ом (3 шт);
- R4 — резистор на 27 Ом;
- R5 — резистор на 68 Ом;
- R6, R7 — резисторы на 100 кОм (2 шт);
- VR1, VR2 — горизонтальные подстроечные резисторы на 100 кОм (2 шт.), необязательные;
- транзисторный зуммер;
- полоса на 3, 4, 5 и 8 гнезд;
- плата с полосками: 12 полос x 30 отверстий.

Внеплатные компоненты:
- D1–D3 — светодиоды диаметром 5 мм, яркие (3 шт.);
- LDR1 и LDR2 — фоторезисторы (типа ORP12 или подобные);
- S1 — малый двухпозиционный переключатель типа SPST;
- батарейный отсек 4 x AAA или 4 x AA с проволочными или штифтовыми выводами;
- соединитель батареи типа PP3 (если отсек — со штифтовыми выводами);
- перезаряжаемые NiMH-элементы AAA или AA (4 шт.).

Разное:
- лента с самоклеющейся тыльной стороной (например, Velcro);
- одножильный монтажный провод (соответствующий диаметру гнезд);
- припой.

## Программирование на ассемблере

В данном разделе предполагается, что при написании программ на ассемблере используется программатор PICkit 2 и его программное обеспечение. В случае использования другого программатора листинг программы будет, в основном, таким же, однако — с небольшими отличиями. Для разработчиков на PICBASIC соответствующие версии программ представлены в следующем разделе книги.

### Сообщение "Hello World!"

Стало уже почти традицией, что в учебниках по программированию все начинается с простейшей программы, отображающей на экране монитора сообщение "Hello World" ("Привет, мир!"). Рассмотренная далее программа — ее робототехнический эквивалент. Она заставляет робот "Скутер" продемонстрировать свои возможности ввода-вывода под управлением простейшей подпрограммы. В то же время, это — способ проверки работоспособности схем вывода данных.

Первое, что делает робот "Скутер" при запуске программы, — остается неподвижным в течении примерно пяти секунд, ожидая внимания окружающих. Затем он на пять секунд включает два передних светодиода, после чего движется вперед в течение двух секунд (убедитесь, что вы не стоите на его пути). В заключение "Скутер" на две секунды включает свой боковой светодиод.

Программа, выполняющая эти действия, показана в листинге 6.1, а соответствующий ей HEX-файл — в листинге 6.2.

Соответствующие файлы `Scoot01.asm` и `Scoot01.hex` можно также найти на прилагаемом к книге компакт-диске в папке `Скутер`.

**Листинг 6.1. Файл `Scoot01.asm`**

```
;*****************************************************************
; Имя файла: Scoot01.asm                                         *
;                                                                *
; Hello World!                                                   *
;                                                                *
;*****************************************************************

  list      p=16F690     ; Микроконтроллер
  __CONFIG  0x30c4

; Банк0
  status    equ 03h
  portb     equ 06h
```



**Листинг 6.1. Продолжение**

```
  portc     equ 07h
; Банк1
  trisb     equ 06h
  trisc     equ 07h
; Банк2
  ansel     equ 1eh
  anselh    equ 1fh
; Код пункта назначения
  f         equ 01h
; Метки
  delay0    equ 20h
  delay1    equ 21h
  delayn    equ 22h

  org 00h
  goto start
  org 04h
  goto start

start
  bcf status, 5     ; Банк0
  bcf status, 6
  clrf portb
  clrf portc
  bsf status, 5     ; Банк1
  clrf trisb        ; Все выводы порта B - входы
  clrf trisc        ; Все выводы порта C - выходы
  bcf status, 5     ; Банк2
  bsf status, 6
  clrf ansel        ; Цифровой ввод-вывод
  clrf anselh       ; Цифровой ввод-вывод
  bcf status, 6     ; Банк0
  bcf status, 5

; Программа начинается здесь
  clrf portb
  clrf portc
  movlw 019h        ; Задержка 5 с. Готовьтесь наблюдать
  call longdelay

  bsf portc, 4      ; Включение фар
  movlw 019h        ; Задержка 5 с
  call longdelay
  bcf portc, 4

  bsf portc, 7      ; Движение вперед
```

**Листинг 6.1. Окончание**

```
  movlw 0ah          ; Задержка 2 с
  call longdelay
  clrf portc         ; Останов

  bsf portc, 5       ; Включение бокового светодиода
  bsf portb, 6       ; Включение зуммера
  movlw 019h         ; Задержка 2 с
  call longdelay
  clrf portc         ; Выключение бокового светодиода
  clrf portb         ; Выключение зуммера

endit  goto endit

; Подпрограммы

delay
  decfsz delay0, f
  goto delay
  decfsz delay1, f
  goto delay
  return

longdelay
  movwf delayn
repeat
  call delay
  decfsz delayn, f
  goto repeat
  return

end
```

**Листинг 6.2. Файл Scoot01.hex**

```
:020000040000FA
:020000000528D1
:0800080005288312031386019
:1000100087018316860187018312031 79E019F01C2
:1000200003138312860187011930 2A20071619301D
:100030002A20071287170A302A2087018716061 7F9
:1000400019302A2087018601242 8A00B2528A10B1E
:0E00500025280800A2002520A20B2B2808005E
:02400E00C430BC
:00000001FF
```



Программа `Scoot01.asm` начинается с идентифицирующей "шапки", которая игнорируется ассемблером, поскольку ассемблер всегда игнорирует то, что указано в командной строке справа от символа ";". Этот заголовок можно изменить или вообще убрать его.

Обычно первые строки любой программы определяют переменные и другие начальные настройки. Так, в листинге 6.1 объявляется тип микроконтроллера, для которого написана программа (в данном примере — PIC16F690). Далее следует конфигурационный код `0x33c4`. Обратите внимание на то, что директива `__CONFIG` начинается двумя символами подчеркивания.

Затем листинг продолжается директивами, которые назначают адреса регистрам, определяют кодовые значения `w` (рабочий регистр) и `f` (текущий файловый регистр) и присваивают метки подпрограммам задержки. В зависимости от используемого программного обеспечения, может оказаться, что все эти адреса указывать необязательно, поскольку они уже встроены в программную оболочку или задаются с помощью включаемого файла.

Теперь настроим микроконтроллер таким образом, чтобы он правильно взаимодействовал с выходными схемами (входы от фоторезисторов в данной программе не используются). Разряды <5> и <6> регистра STATUS переключают банки регистров специального назначения. Регистры для порта B и порта C в банке 0 обнуляются. Порт A в этой программе не используется. Далее программа переключается на банк 1, чтобы установить регистры трех состояний таким образом, чтобы все каналы портов B и C были выходными.

В отличие от некоторых более ранних моделей микроконтроллеров PIC, каналы 16F690 являются аналоговыми по умолчанию и должны быть переведены в цифровой режим явным образом. Для этого следует переключиться на банк 2, где размещены регистры ANSEL и ANSELH, и обнулить все их разряды.

Фактическая программа начинается с обнуления регистров порта, чтобы гарантировать, что светодиоды не включатся сразу же. После этого реализована временная задержка с помощью подпрограммы `longdelay`.

Светодиоды и двигатель включаются и выключаются путем установки разрядов в регистрах порта. Устанавливаемые разряды указаны в табл. 6.2. Так, например, установка разряда 4 порта C переводит вывод RC4 в состояние лог. 1, что замыкает транзисторный ключ, включающий светодиоды LED1 и LED2. Команда `bsf` устанавливает разряд в `1`, а команда `bcf` — обнуляет. Наконец, для того, чтобы убедиться, что ни-

что не осталось включенным, с помощью команды `clrf` обнуляются все каналы обоих портов. После этого робот "Скутер" входит в бесконечный цикл `endit`.

Для формирования пауз программа обращается к подпрограмме `longdelay`, которая в свою очередь обращается к подпрограмме `delay`. Обе подпрограммы находятся в конце листинга. Завершается листинг существенно важной директивой `end`.

### Нечто большее, чем просто привет

Рассмотренный листинг можно модифицировать для расширения круга возможностей робота "Скутер":

- мигнуть светодиодами два или три раза, вместо одного;
- подать тройной звуковой сигнал до и после того, как робот проедет вперед;
- проехать назад, а затем вперед в различных направлениях;
- завершить подпрограмму миганием светодиодов в бесконечном цикле.

### Поиск света

В этой программе предполагается, что робот "Скутер" находится в комнате с приглушенным освещением, в которой присутствует один более яркий источник света (например, установленная на полу настольная лампа). Задача робота заключается в том, чтобы локализовать этот источник света и начать движение к нему.

Эта программа также демонстрирует использование компараторов микроконтроллера PIC. Прежде чем перейти к разбору программы поиска света, рассмотрим программу для настройки переменного резистора VR1. Подобные диагностические программы полезны для проверки работоспособности датчика и корректности опорного напряжения.

Программа калибровки компаратора показана в листинге 6.3, а соответствующий ей HEX-файл — в листинге 6.4.

Соответствующие файлы `Scoot02.asm` и `Scoot02.hex` можно также найти на прилагаемом к книге компакт-диске в папке `Скутер`.

**Листинг 6.3. Файл `Scoot02.asm`**

```
;*****************************************************************
; Имя файла: Scoot02.asm                                         *
;                                                                *
; Калибровка компаратора                                         *
;                                                                *
;*****************************************************************
```



**Листинг 6.3. Продолжение**

```
  list       p=16F690     ; Определяем процессор
  __CONFIG 0x30c4

; Банк0
  status     equ 03h
  porta      equ 05h
  portb      equ 06h
  portc      equ 07h
  intcon     equ 0bh
; Банк1
  option_reg equ 01h
  trisa      equ 05h
  trisb      equ 06h
  trisc      equ 07h
; Банк2
  cm1con0    equ 19h
  ansel      equ 1eh
  anselh     equ 1fh
; Коды пунктов назначения
  w          equ 00h
  f          equ 01h
  z          equ 02h
; метки
  delay0     equ 20h
  delay1     equ 21h

  org 00h
  goto start
  org 04h
  goto start

start
  bcf intcon, 7      ; Запрет прерываний
  bcf status, 5      ; Банк0
  bcf status, 6
  bsf status, 5
  clrf trisb         ; Все каналы порта B - выходы
  clrf portc         ; Все каналы порта C - выходы
  bcf status, 5      ; Банк2
  bsf status, 6
  movlw 03h          ; Цифровой вход, за исключением RA0 и RA1
  movwf ansel
  clrf anselh        ; Цифровой выход
  movlw 080h         ; Включение компаратора
  movwf cm1con0
  bcf status, 6      ; Банк0
```

**Листинг 6.3. Окончание**

```
  bcf status, 5

; Программа начинается здесь

  clrf porta
  clrf portb
  clrf portc

sample
  bsf status, 6     ; Банк 2
  movlw 050h        ; Разряд 6 = 1
  andwf cm1con0, w  ; Считывание выходного бита
  bcf status, 6     ; Банк 0
  btfss status, z   ; Проверка флага нулевого результата
  goto sample
  bsf portc, 5      ; Включение бокового светодиода
  call delay
  bcf portc, 5      ; Выключение светодиода
  goto sample

; Подпрограмма

delay
  decfsz delay0, f
  goto delay
  decfsz delay1, f
  goto delay
  return

end
```

**Листинг 6.4. Файл Scoot02.hex**

```
:020000040000FA
:020000000528D1
:0800080005288B138316031376
:10001000860187018312031703309E009F01803001
:10002000990003138312850186018701031750305D
:1000300019050313031D162807142020071016287E
:0A004000A00B2028A10B20280800C7
:02400E00C430BC
:00000001FF
```

После объявления типа PIC и конфигурационного слова определяется список меток для регистров специального назначения. В общем случае этот список излишен, если применяется программная среда разработки, использующая включаемые файлы. В данном случае нет никаких подпрограмм обслуживания прерываний, поэтому мы переходим прямо к главной программе (рис. 6.15).



Если фоторезистор принимает яркий свет, то на компаратор подается высокое входное напряжение. Если это напряжение превышает опорное, поступающее от VR1, то разряд <6> регистра CM1CON0 принимает значение 1. Состояние этого разряда контролируется путем опроса флага нулевого результата в регистре STATUS. Если входное напряжение является высоким, то разряд CM1CON0 <6> содержит 0, и установлен флаг нулевого результата. Команда `btfss status, z` приводит к пропуску следующей командной строки и включению светодиода на 0,2 с. Затем программа вновь возвращается к опросу флага нулевого результата. Если уровень освещенности фоторезистора будет низким, то разряд <6> содержит 1, флаг нулевого результата сброшен, и программа вернется обратно в начало цикла без включения светодиода.


**Рис. 6.15.** Блок-схема показывает, что программа работает в режиме непрерывного цикла. Всякий раз при прохождении цикла опрашивается выход компаратора, а светодиод включается на 0,2 с или не включается, в зависимости от результатов опроса

Разместите робот так, чтобы на него падал свет, столь же яркий, как и лампа-цель, которая будет использоваться при запуске программы поиска света. Запустите программу калибровки. Отрегулируйте VR1 отверткой так, чтобы светодиод срабатывал точно при заданной освещенности. Светодиод должен немедленно гаснуть, если источник света удаляется или же его яркость уменьшается.

Теперь переходим к программе поиска света. Она основана на простой процедуре. Комната заполнена неярким рассеянным светом. В ней присутствует один яркий источник света (например, настольная лампа), расположенный на полу. Робот "Скутер" находится на полу таким образом, чтобы не быть ориентированным на источник света. Дальнейшие события иллюстрирует блок-схема на рис. 6.16.

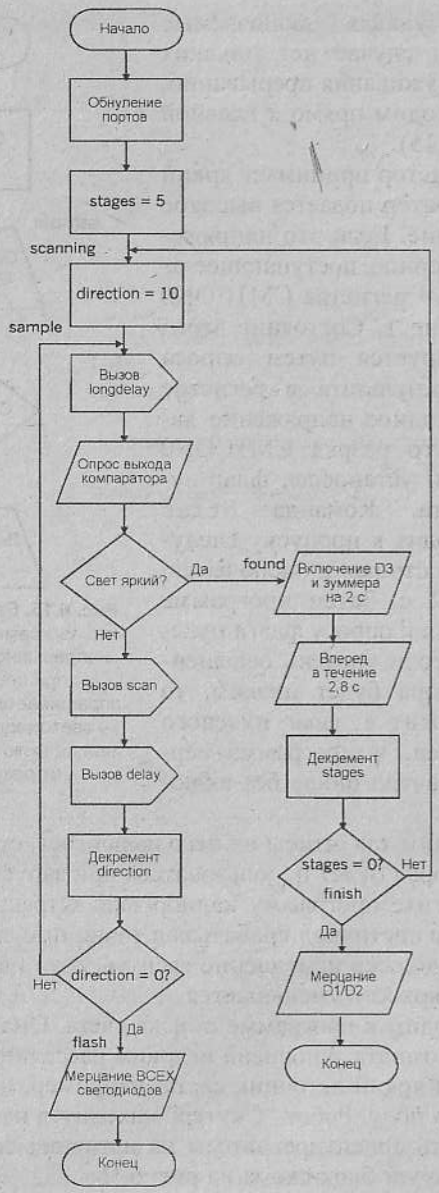



**Рис. 6.16.** Программа поиска света предоставляет роботу пять попыток подобраться к яркому источнику света



Робот "Скутер" вначале пытается найти источник света с помощью подпрограммы, которая начинается с метки `sample`. Он опрашивает выход компаратора, чтобы определить, является ли источник света перед ним ярким или неярким. Под словом "яркий" мы подразумеваем, что напряжение с фоторезисторного датчика превышает опорное.

Если свет перед роботом недостаточно яркий, то "Скутер" переходит к выполнению подпрограммы `scan`. Он немного отъезжает назад, поворачивая вправо по мере движения, после чего немного продвигается вперед. Фактически, он немного повернулся вправо и теперь ориентирован в другом направлении. До тех пор, пока источник яркого света не попадет в его поле зрения, робот будет продолжать опрашивать выход компаратора и немного поворачиваться.

После 10 таких попыток, соответствующих осмотру комнаты на 360°, он придет к выводу, что яркий источник света не обнаружен. Робот прекратит процесс поиска и начнет мигать светодиодами. На этом работа программы завершается.

Если при опросе выхода компаратора и сканировании, компаратор выдаст лог. 1, то робот будет знать, что в данный момент времени он ориентирован на источник яркого света. Он перейдет к метке `found` программы, где прежде всего отметит свой успех включением бокового светодиода и подачей звукового сигнала. Затем он двинется вперед, по направлению к источнику света.

Однако, угловое разрешение фотоэлемента достаточно большое, поэтому робот может быть сориентирован недостаточно точно. Проехав вперед в течение нескольких секунд, он может немного сбиться с пути, поэтому должен приближаться к источнику света поэтапно, проверяя в конце каждого этапа направление движения. Таким образом, остановившись, робот возвращается к метке `sample` и повторяет этап считывания сигнала и сканирования. Вновь обнаружив свет, он начинает двигаться вперед (возможно, в другом направлении). Таким образом, достижение лампы происходит в пять этапов. По их прошествии робот остановится и начнет мигать передними светодиодами.

Программа поиска света показана в листинге 6.5, а соответствующий ей HEX-файл — в листинге 6.6.

Соответствующие файлы `Scoot03.asm` и `Scoot03.hex` можно также найти на прилагаемом к книге компакт-диске в папке `Скутер`.

**Листинг 6.5. Файл `Scoot03.asm`**

```
;*****************************************************************
; Имя файла: Scoot03.asm
```

**Листинг 6.5. Продолжение**

```
; Робот Scooter ищет свет, используя компаратор.          *
;                                                         *
;*********************************************************

  list      p=16F690     ; Определение процессора
  __CONFIG  0x30c4

; Банк0
  status    equ 03h
  porta     equ 05h
  portb     equ 06h
  portc     equ 07h
  intcon    equ 0bh
; Банк1
  trisa     equ 05h
  trisb     equ 06h
  trisc     equ 07h
; Банк2
  cm1con0   equ 19h
  ansel     equ 1eh
  anselh    equ 1fh
; Биты
  w         equ 00h
  f         equ 01h
  z         equ 02h
; Метки
  delay0    equ 20h
  delay1    equ 21h
  direction equ 23h
  stages    equ 24h
  org 00h
  goto start
  org 04h
  goto start

start
  bcf intcon, 7     ; Запрет прерываний
  bcf status, 5     ; Банк0
  bcf status, 6
  bsf status, 5
  clrf trisb        ; Все разряды порта B - выходы
  clrf portc        ; Все разряды порта C - выходы
  bcf status, 5     ; Банк2
  bsf status, 6
  movlw 03h         ; Цифровые входы, за исключением RA0 и RA1
  movwf ansel
```



**Листинг 6.5. Продолжение**

```
  clrf anselh       ; Цифровой выход
  movlw 080h        ; Включение компаратора
  movwf cm1con0
  bcf status, 6     ; Банк0
  bcf status, 5

; Программа начинается здесь

  clrf porta
  clrf portb
  clrf portc
  movlw 08h
  movwf stages
scanning
  movlw 0ah         ; direction = 10
  movwf direction
sample
  movlw 03h
  call longdelay
  bsf status, 6     ; Банк 2
  movlw 050h        ; Разряд 6 = 1
  andwf cm1con0, w  ; Считывание выходного бита
  bcf status, 6     ; Банк 0
  btfss status, z   ; Проверка флага нулевого результата
  goto found
  call scan
  call delay
  decfsz direction  ; Обратный отсчет сканирований
  goto sample       ; Повтор попытки
  goto flash
found
  bsf portc, 5      ; Включение D3
  bsf portb, 6      ; Включение зуммера
  movlw 0ah
  call longdelay
  clrf portc        ; Выключение D3
  clrf portb        ; Выключение зуммера
  bsf portc, 7      ; Вперед
  movlw 0eh
  call longdelay
  clrf portc        ; Останов
  decfsz stages
  goto scanning
  goto finish
flash
  bsf portc, 5      ; Включение бокового светодиода
```

**Листинг 6.5. Окончание**

```
  bsf portc, 4      ; Включение передних светодиодов
  movlw 03h
  call longdelay
  clrf portc
  movlw 03h
  call longdelay
  goto flash
finish
  bsf portc, 4      ; Включение передних светодиодов
  movlw 03h
  call longdelay
  clrf portc
  movlw 03h
  call longdelay
  goto finish

; подпрограммы
scan
  bsf portc, 6      ; Назад
  movlw 05h
  call longdelay
  clrf portc        ; Останов
  call delay
  bsf portc, 7      ; Вперед
  movlw 05h
  call longdelay
  clrf portc        ; Останов
  return
```

**Листинг 6.6. Файл Scoot03.hex**

```
:020000040000FA
:020000000528D1
:0800080005288B13831203137A
:1000100083168601870183120317033 09E009F0118
:10002000803099000313831285018601870108300F
:10003000A4000A30A3000330532003175030190 5E1
:100040000313031D282844204E20A30B1B2835280A
:1005000871606170A30532087018601871 0E304E
:10060005320870 A40B19283D2887160716033053
:1007000532087010330532035280716033053 0BF
:10080008701033053203D28071705305320870 8F
:100900004E208717053053208701080 A00B4E28FB
:1000A000A10B4E280800A2004E20A20B54280800E5
:02400E00C430BC
:00000001FF
```



Программа в листинге 6.5 подразумевает наличие подпрограмм `delay` и `longdelay`, за которыми следует директива `end`. Она использует внешнее опорное напряжение, что позволяет перенастраивать уровень переключения с помощью отвертки вместо редактирования и повторного ассемблирования программы.

Роботы отличаются своими двигателями и колесам, поэтому значения `10` переменной `direction` может оказаться недостаточно для покрытия всей окружности. В таком случае отредактируйте программу, задав большее значение для `direction` (первая командная строка после метки `scanning`).

Если робот "Скутер" запускается в большом помещении, то ему следует предоставить возможность сделать еще несколько шагов для достижения лампы. Для этого увеличьте значение переменной `stages` (командная строка сразу перед меткой `scanning`).

### Обход препятствий

Для выполнения этого маневра в помещении должно быть достаточно места на полу. Кроме того, она должна бать равномерно освещена. Освещение может быть достаточно ярким при условии, что оно не превзойдет интенсивность света, исходящего от двух передних светодиодов. Рассмотренная далее программа иллюстрирует метод обнаружения препятствий, основанный на анализе освещенности.

Данный метод использует АЦП микроконтроллер PIC для измерения интенсивности света, попадающего на передние фоторезисторы в то время, когда включены передние светодиоды. Эти замеры сравниваются с замерами, полученными с фоторезисторов, когда светодиоды были выключены.

Светодиоды смонтированы на корпусе и направлены на одну центральную точку, расположенную на расстоянии около 100 мм перед роботом. Если в этой точке присутствует объект, то исходя из предположения, что этот объект достаточно большой и хорошо отражает свет, можно утверждать, что количество света, принятого фоторезистором, будет значительно больше, чем обычно. Робот распознает эту ситуацию и начнет выполнять действия, направленные на обход препятствия.

Робот отреагирует также тогда по мере приближения к стене или мебели, в силу чего он мог бы передвигаться по комнате неопределенно долго, хотя в реальности может время от времени натыкаться на углы предметов, как бы "не видя" их. Можете попробовать разработать программу, которая позволила бы роботу избежать подобных "ловушек".

Программа дважды включает подпрограмму записи выходных данных АЦП в память EEPROM. Она была введена для контроля за обработкой данных. После запуска программы с целью тестирования поведения робота, микроконтроллер PIC был возвращен в программатор для сохранения считываемых данных. Эти подпрограммы не являются обязательной частью конечной программы, поэтому их можно не вводить, однако на стадии разработки они очень полезны.

Основной программный цикл относительно прост (рис. 6.17).

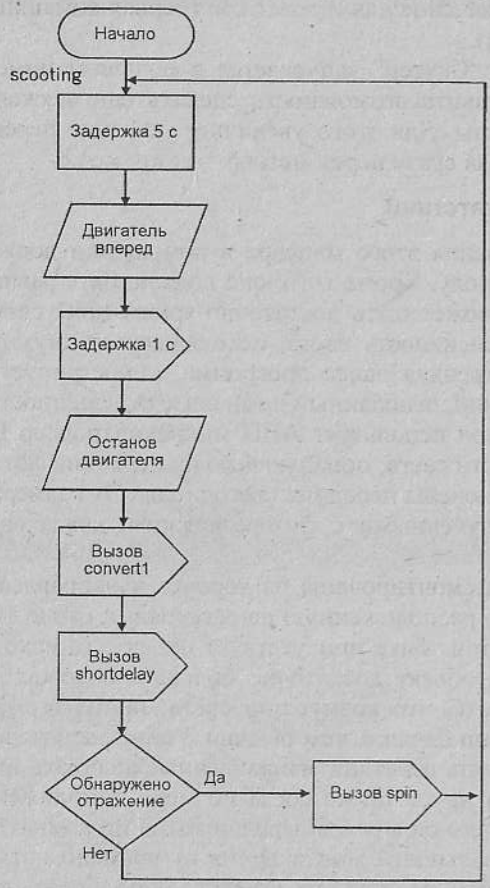



**Рис. 6.17.** В основном цикле программы обхода препятствия робот немного перемещается вперед, а затем включает и выключает светодиоды (в подпрограмме `convert1`). Распознав препятствие, он сдаст назад и повернется (в подпрограмме `spin`). Если же препятствие не обнаружено, то робот продолжает двигаться прямо вперед



Примечательная часть этой программы — подпрограмма `convert1` (рис. 6.18).

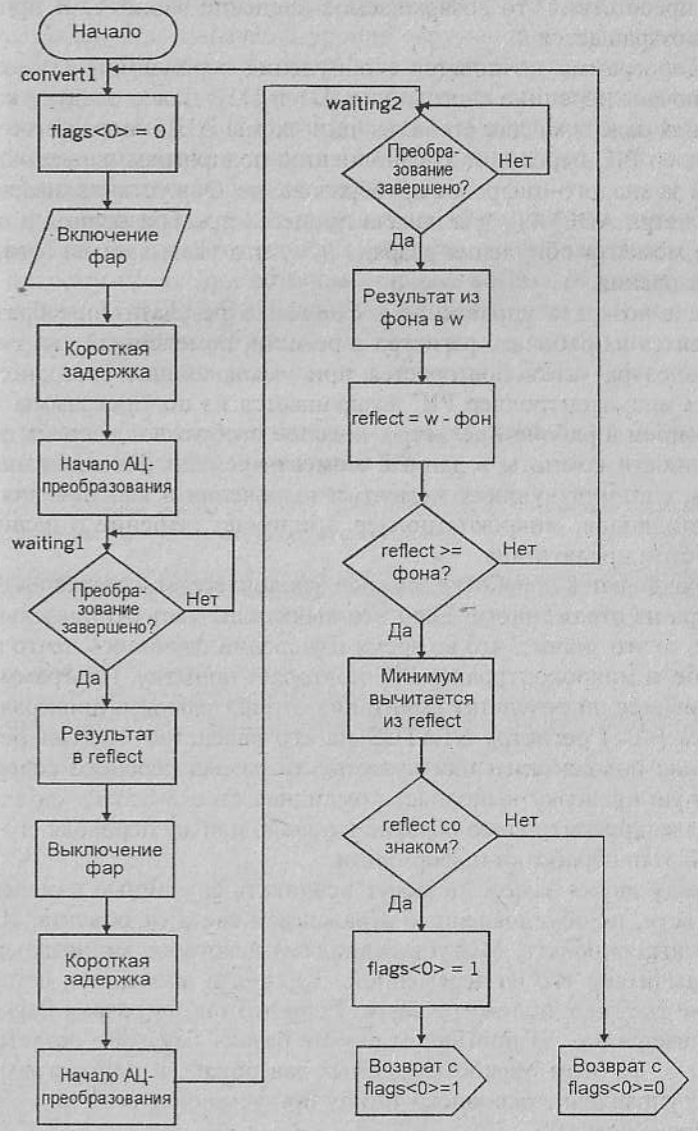



**Рис. 6.18.** Подпрограмма `convert1` программы обхода препятствия

Работа этой подпрограммы заключается в возврате значения, сохраненного в переменной `flags`. Если впереди робота не обнаружено никакого препятствия, то возвращаемое значение равно `1`. В противном случае возвращается `0`.

Подпрограмма начинается с обнуления переменной `flags`. Затем она включает передние светодиоды (D1 и D2). Далее следует короткая временная задержка для стабилизации схемы АЦП, после чего микроконтроллер PIC переходит к выполнению подпрограммы `adread`, отвечающей за аналого-цифровое преобразование. Она устанавливает разряд <1> регистра ADCON0 для начала процесса преобразования и ожидает в цикле момента обнуления разряда <1>, что указывает на завершение преобразования.

После возврата управления к `convert1` результат преобразования переносится из рабочего регистра в регистр, помеченный как `reflect`. Вся процедура затем повторяется при выключенных светодиодах. На этот раз микроконтроллер PIC возвращается из подпрограммы `adread` со значением в рабочем регистре, которое отображает уровень фоновой освещенности комнаты в данный момент времени. На основании двух замеров, соответствующих моментам включения и выключения передних светодиодов, микроконтроллер принимает решение о наличии перед роботом препятствия.

Первый шаг в обработке замеров заключается в вычитании фонового замера из отраженного. Если это вычитание дает отрицательный результат, то это значит, что во время измерений случилось нечто непредвиденное, и микроконтроллер PIC повторяет попытку. Программа выясняет, является ли результат вычитания отрицательным, проверяя разряд переноса (<0>) регистра STATUS на его равенство `0`. Если результат вычитания положителен или нулевой, то разряд переноса содержит `1`. Названную проверку выполняет командная строка `btfss status, 0`, посылая микроконтроллер обратно в начало или же переводя его на следующий этап обработки информации.

Между двумя замерами могут возникать случайные изменения освещенности, не обусловленные отражением света от объекта. Их необходимо игнорировать. Мы устанавливаем некоторое минимальное значение, вычитаем его из переменной `reflect` и проверяем, остается ли значение `reflect` положительным. Если это так, то разряд flags<0> устанавливается в `1`. В противном случае разряд flags< 0> остается в состоянии `0`. На этом обработка данных завершена, и подпрограмма возвращает управление основному циклу при установленном или сброшенном разряде flags<0>.



Величина, вычитаемая из `reflect`, управляет чувствительностью обработки данных. Если ее выбрать слишком большой, то робот будет пропускать объекты, если же — слишком малой, то робот будет реагировать на небольшие изменения уровня окружающей освещенности, не связанные с наличием или отсутствием объекта перед ним. Если окажется, что вычитаемое значение для вашего робота некорректно, отредактируйте программу. Для этого и предназначены подпрограммы, которые сохраняют данные в памяти EEPROM: они позволяют считывать значения, возвращаемые при двух замерах.

В данной программе мы не используем 10-разрядную точность, поэтому преобразователь был сконфигурирован на выравнивание результата по правому краю, при котором старшие восемь разрядов помещаются в регистр ADRESH. Это делает командная строка `bsf eecon1, 7`. После возврата из подпрограммы `adread` результат будет в регистре ADRESH, а два младших разряда — в регистре ADRES (они игнорируются).

Программа обхода препятствий показана в листинге 6.7, а соответствующий ей HEX-файл — в листинге 6.8.



Соответствующие файлы `Scoot04.asm` и `Scoot04.hex` можно также найти на прилагаемом к книге компакт-диске в папке `Скутер`.

**Листинг 6.7. Файл `Scoot04.asm`**

```
;*****************************************************************
;                                                                *
;    Имя файла: Scoot04.asm                                      *
;    "Скутер" обходит препятствия.                               *
;    Применение АЦП. Используйте EEPROM для мониторинга.         *
;                                                                *
;*****************************************************************

  list       p=16F690            ; Определение процессора
  __CONFIG   0x30c4

; Банк0
  status     equ 03h
  porta      equ 05h
  portb      equ 06h
  portc      equ 07h
  intcon     equ 0bh
  adresh     equ 1eh
  adcon0     equ 1fh
; Банк1
  trisa      equ 05h
```

**Листинг 6.7. Продолжение**

```
  trisb       equ 06h
  trisc       equ 07h
  adcon1      equ 1fh
; Банк2
  eedat       equ 0ch
  eeadr       equ 0dh
  ansel       equ 1eh
  anselh      equ 1fh
; Банк3
  eecon1      equ 0ch
  eecon2      equ 0dh
; Разряды
  w           equ 00h
  f           equ 01h
  z           equ 02h
; Метки
  delay0      equ 20h
  delay1      equ 21h
  delayn      equ 22h
  reflect     equ 23h
  flags       equ 24h

  org 00h
  goto start
  org 04h
  goto start

start
  bcf intcon, 7      ; Запрет прерываний
  bcf status, 5      ; Банк0
  bcf status, 6
  clrf porta
  clrf portb
  clrf portc
  bsf status, 5      ; Банк1
  clrf trisb         ; Все выводы порта B - выходы
  clrf portc         ; Все выводы порта C - выходы
  bcf status, 5      ; Банк2
  bsf status, 6
  movlw 03h          ; Цифровой вход, за исключением RA0 и RA1
  movwf ansel
  clrf anselh        ; Цифровой выход
  bcf status, 6      ; Банк0
  bcf status, 5

; Программа начинается здесь
```



**Листинг 6.7. Продолжение**

```
  movlw 019h         ; Задержка 5 с
  call longdelay

scooting
  call shortdelay
  bcf flags, 0
  bcf portc, 6       ; Двигатель вперед
  bsf portc, 7
  movlw 05h
  call longdelay

  bcf portc, 7       ; Останов
  call convert1      ; Опрашиваем передний датчик
  call shortdelay
  btfss flags, 0     ; Если есть отражение
  goto scooting
  call spin
  goto scooting

flash
  bsf portc, 5       ; Включаем боковой светодиод
  call delay
  bcf portc, 5       ; Выключаем боковой светодиод
  call delay
  goto flash

; Подпрограммы

shortdelay
  movlw 060h         ; Установка длительности задержки
  movwf delay1
  call delay
  clrf delay1        ; Восстанавливаем delay1
  return
delay
  decfsz delay0, f
  goto delay
  decfsz delay1, f
  goto delay
  return
longdelay
  movwf delayn
repeat
  call delay
  decfsz delayn, f
  goto repeat
```

**Листинг 6.7. Продолжение**

```
  return
convert1
  bcf flags, 0      ; Сбрасываем флаг отражения
  bsf portc, 4      ; Включаем фары
  call shortdelay
  call adread
  movf adresh, w    ; Считываем 8-разрядный результат в w
  movwf reflect     ; Отраженный свет

; Запись данных в EEPROM (необязательная)

  bsf status, 6     ; Банк2
  bcf status, 5
  movlw 01h         ; Сохранен адрес
  movwf eeadr
  bcf status, 6     ; Банк0
  movf adresh, w    ; Данные
  bsf status, 6     ; Банк2
  movwf eedat
  bsf status, 6
  bsf status, 5     ; Банк3
  bcf eecon1, 7
  bsf eecon1, 2     ; Разрешаем запись данных
  movlw 055h
  movwf eecon2
  movlw 0aah
  movwf eecon2
  bsf eecon1, 1     ; Начало процесса записи
notdone
  btfsc eecon1, 1
  goto notdone
  bcf eecon1, 2     ; Запрещаем запись
  bcf status, 6     ; Банк0
  bcf status, 5
  bcf portc, 4      ; Выключаем фары
  call shortdelay
  call adread

; Запись данных в EEPROM (необязательная)

  bsf status, 6
  bcf status, 5
  movlw 02h
  movwf eeadr
  bcf status, 6
  movf adresh, w
```



**Листинг 6.7. Продолжение**

```
  bsf status, 6
  movwf eedat
  bsf status, 6
  bsf status, 5
  bcf eecon1, 7
  bsf eecon1, 2
  movlw 55h
  movwf eecon2
  movlw 0aah
  movwf eecon2
  bsf eecon1, 1
notdone1
  btfsc eecon1, 1
  goto notdone1
  bcf eecon1, 2
  bcf status, 6
  bcf status, 5
  movf adresh, w    ; Считываем 8-разрядный результат в w
  subwf reflect, f  ; Отраженный - фоновый
  btfss status, 0   ; Отраженный >= фоновый
  goto convert1     ; Ложный замер - попробуем еще раз
  movlw 19h         ; Минимальнео значимое различие
  subwf reflect, w
  btfss status, 0
  return            ; Отражение не обнаружено
  bsf flags, 0      ; Отражение - устанавливаем флаг
  return            ; Отражение обнаружено

adread
  movlw 050h        ; Выбираем тактовую частоту 1/16 Fosc
  movwf adcon1
  movlw 01h         ; Выравнивание по левому краю,
                    ; опорное напряжение - напряжение питания,
                    ; канал 0, активизация
  movwf adcon0
  call delay        ; Для опроса напряжения
  bsf adcon0, 1     ; Начало преобразования
waiting
  call delay
  btfsc adcon0, 1   ; Если преобразование завершено
  goto waiting
  return
spin
  bsf portb, 6      ; Включаем зуммер
  bsf portc, 5
  call delay
```

**Листинг 6.7. Окончание**

```
  bsf portc, 6      ; Назад и поворачиваем
  bcf portc, 7
  movlw 05h
  call longdelay
  bcf portc, 6      ; Останов
  bcf portb, 6      ; Выключаем зуммер
  bcf portc, 5
  return

end
```

**Листинг 6.8. Файл Scoot04.hex**

```
:020000040000FA
:020000000528D1
:0800080005288B13831203137A
:1000100085018601870183168601870183120317F4
:1000200003309E009F0103138312193033202920CF
:1000300024100713871705303320871338202920 11
:10040000241C17288120172887162E2087122E207F
:1005000024286030A1002E20A1010800A00B2E282A
:10060000A10B2E280800A2002E20A20B3428080085
:1007000024100716292077201E08A30003178312D7
:1008000001308D0003131E0803178C00031783161D
:100900008C130C1555308D00AA308D008C148C18E3
:1009A0004F280C11031383120712292077200317FE
:1009B000831202308D0003131E0803178C000317F0
:1009C00083168C130C1555308D00AA308D008C14BE
:1009D0008C1868280C11031383121E08A302031C3A
:1009E000382819302302031C08002414080050305B
:1009F0009F0001309F002E209F142E209F187D28E6
:100A0000080006178716 2E2007178713053033209F
:080110000713061387120800 13
:02400E00C430BC
:00000001FF
```

Обработка аналоговых данных, доступная в микроконтроллере 16F690, — это еще не все, что может сделать робот "Скутер", если его оснастить дополнительными датчиками или исполнительными устройствами. Так, например, можно добавить звуковой датчик и запрограммировать его на срабатывание по щелчку пальцами. Или же можно установить спереди инфракрасный датчик, ориентировав его вниз, как в роботе "Искатель" (см. главу 9). Затем его можно запрограммировать так, чтобы робот избегал пересечения черных линий. Его можно будет "поймать в западню", просто обведя вокруг него черную линию. Если



же реализовать смещение влево, то робот можно запрограммировать на отслеживание линии или на проход лабиринта.

Более амбициозный проект — создать два робота "Скутер", которые могли бы общаться друг с другом через мигание светодиодов или по радиоканалу. Совместное поведение двух роботов — интересный предмет для исследований.

## Программирование на PICBASIC

В данном разделе описаны программы на языке BASIC, аналогичные рассмотренным выше ассемблерным программам. Используйте их, а также — представленные блок-схемы для перевода ассемблерного кода в BASIC-программы.

### Сообщение "Hello World!"

Программа "Hello World!" на языке PICBASIC представлена в листинге 6.9.

Соответствующий файл Scoot01.txt можно также найти на прилагаемом к книге компакт-диске в папке Скутер\PICBASIC.

**Листинг 6.9. Файл Scoot01.txt**

```
' Scoot01.txt
Symbol Portb = $6
Symbol Portc = $7
Symbol Trisb = $86
Symbol Trisc = $87
Symbol Ansel = $11E
Symbol Anselh = $11F

  Poke Trisb, 0      ' Все каналы порта B - выходы
  Poke Trisc, 0      ' Все каналы порта C - выходы
  Poke Ansel, 0      ' Цифровой ввод-вывод
  Poke Anselh, 0     ' Цифровой ввод-вывод

' Программа начинается здесь
  Poke Portb, 0
  Poke Portc, 0
  Pause 5000         ' Задержка 5 с
  Poke Portc, $10    ' Включение передних светодиодов
  Pause 5000         ' Задержка 5 с
  Poke Portc, $80    ' Движение вперед
  Pause 2000         ' Задержка 2 с
  Poke Portc, $20    ' Включение бокового светодиода
  Poke Portb, $40    ' Включение зуммера
```

**Листинг 6.9. Окончание**

```
  Pause 2000          ' Задержка 2 с
  Poke Portc, 0       ' Выключение бокового светодиода
  Poke Portb, 0       ' Выключение зуммера
End
```

Очевидно, что по сравнению с листингом 6.1 программа на BASIC намного короче. В общем случае это справедливо для всех BASIC-версий ассемблерных программ. Язык BASIC предоставляет встроенные подпрограммы, решающие сложные задачи, в то время как ассемблеру необходимо указывать, что делать, шаг за шагом. Тем не менее, в машинных кодах ассемблерные программы в общем случае намного короче, чем их версии, сгенерированные компилятором BASIC.

Примечательной чертой листинга 6.9 является высокая пропорция операторов `Poke`. Язык PICBASIC для перевода выходов в состояние лог. 1 или лог. 0 обычно использует команды `High` и `Low`, которые применимы только к разрядам порта B. По различным причинам, микроконтроллер PIC робота "Скутер" настроен на использование порта C для большинства операций вывода данных. Переделка его на использование порта B означала бы изменение разводки проводов. Во избежание этого мы использовали операторы `Poke`. Кроме того, порт B в микроконтроллере 16F690 предоставляет только четыре разряда. В ассемблерных версиях программ мы можем устанавливать или сбрасывать различные разряды, однако оператор `Poke` воздействует на все разряды порта одновременно. Существуют способы поправить это. О них мы расскажем позже.

Аналогичным образом, команда `Peek` одновременно считывает все разряды порта.

### Поиск света

Версия этой программы на PICBASIC достигает той же цели, что и ассемблерная версия, но другим способом. В случае с ассемблером робот "Скутер" кружит по комнате до тех пор, пока не обнаружит источник яркого света. После этого он начинает движение на свет. В версии программы на BASIC робот поворачивается по крайней мере на 360°, измеряя по ходу уровень освещенности. Он определяет, какой из возможных нескольких источников света — наиболее яркий, и затем двигается к нему. Соответствующая блок-схема показана на рис. 6.19.

В данном случае робот "Скутер" использует вместо компаратора один из своих АЦП. Это означает, что у него нет необходимости использовать вывод 19 (AN0) в качестве входа опорного напряжения



(VR1). Мы можем просто игнорировать этот вход или же использовать его в качестве альтернативного входа второго АЦП. Вывод 19 микроконтроллера PIC (RA0/AN0) на F12 соединен с резистором R7 в точке E1. Это обеспечивает возможность использования LDR2 в качестве фотоэлемента на левой стороне робота.

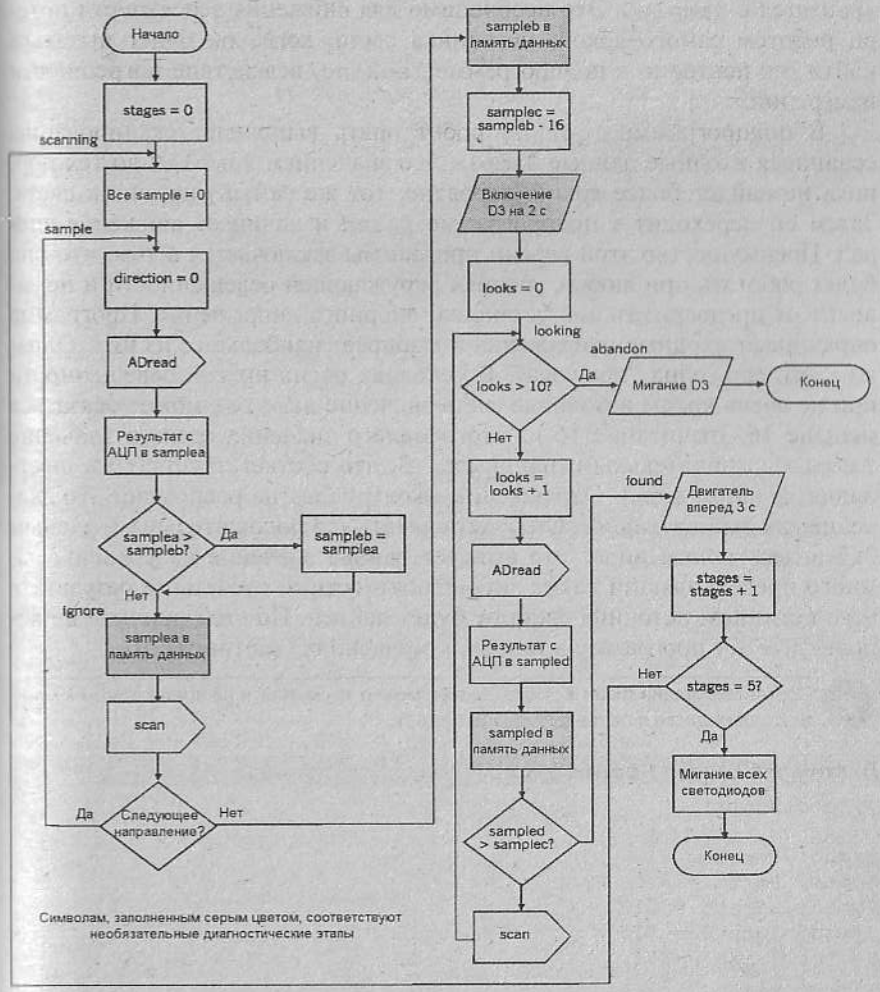



**Рис. 6.19.** Версия программы на языке BASIC включает в себя подпрограмму для обнаружения максимума освещенности

Программа содержит подпрограмму поиска самого яркого источника света. По мере сканирования и отбора данных последовательные замеры, считываемые в переменную SampleA, сравниваются с замером в переменной SampleB и больший из них заносится в SampleB. В конце фазы выборки данных значение SampleB немного уменьшается и сохраняется в SampleC. Это необходимо для снижения вероятности потери роботом самого яркого источника света, когда он будет пытаться найти его повторно в подпрограмме looking, вследствие погрешности измерений.

В подпрограмме looking робот опять выполняет сканирование, сравнивая входные данные SampleD со значением SampleC до тех пор, пока не найдет более яркий (вероятно, тот же самый) источник света. Затем он переходит к подпрограмме found и начинает движение вперед. Преимущество этой версии программы заключается в том, что она будет работать при любых уровнях окружающей освещенности и не зависит от предварительной установки опорного напряжения. Программа опрашивает входные напряжения и выбирает наибольшее из них. Однако здесь есть одна "ловушка". В условиях очень низкой освещенности при не очень ярком источнике света значение SampleB может оказаться меньше 16. Вычитание 16 из этого малого значения сделает значение SampleC отрицательным (например, –3, что соответствует $fd в шестнадцатеричном виде). Однако, микроконтроллер не распознает это значение как отрицательное. Он будет считать его положительным, равным 253 в десятичном виде. Это выведет данное значение на уровень, намного превосходящий какие либо вероятностные отсчеты, в результате чего истинный источник света не будет найден. По этой причине не используйте эту программу в темных помещениях (листинг 6.10).



Соответствующий файл Scoot03.txt можно также найти на прилагаемом к книге компакт-диске в папке Скутер\PICBASIC.

**Листинг 6.10. Файл Scoot03.txt**

```
' Scoot03.txt

Symbol Portb = $6
Symbol Portc = $7
Symbol Adresh = $1E
Symbol Adcon0 = $1F
Symbol Trisb = $86
Symbol Trisc = $87
Symbol Adcon1 = $9F
Symbol Ansel = $11E
Symbol Anselh = $11F
```



**Листинг 6.10. Продолжение**

```
Symbol SampleA = B2
Symbol SampleB = B3
Symbol SampleC = B4
Symbol SampleD = B5
Symbol Looks = B6
Symbol Finds = B7
Symbol Direction = B8
Symbol Stages = B9
  Poke Trisb, 0      ' Все разряды порта B - выходы
  Poke Trisc, 0      ' Все разряды порта C - выходы
  Poke Ansel, 3      ' Цифровые входы, за исключением RA0 и RA1
  Poke Anselh, 0
  Poke Portb, 0
  Poke Portc, 0
  Pause 4000
  Stages = 0
Scanning:
  SampleA = 0
  SampleB = 0
  SampleC = 0
  SampleD = 0
Sample:
  For Direction = 1 to 10
  Pause 500                ' Разрешение установки времени
  Gosub ADread
  Peek Adresh, SampleA     ' Считывание результата в SampleA
  If SampleA < SampleB Then Ignore      ' Находим самый яркий
  SampleB = SampleA
Ignore:
  Write Direction, SampleA
  Gosub Scan
  Next Direction
  Write 11, SampleB
  SampleC = SampleB - $10  ' Немного уменьшаем
  Poke Portc, $20          ' D3
  Pause 2000
  Poke Portc, 0            ' Выключаем D3
  Looks = 0
Looking:
  If Looks > 10 Then Abandon
  Looks = Looks + 1
  Pause 500
  Gosub ADread             ' Поиск яркого источника света
  Peek Adresh, SampleD
  Write 12, SampleD
  If SampleD > SampleC Then Found       ' Яркий источник найден
```

**Листинг 6.10. Продолжение**

```
  Gosub Scan
  Goto Looking
Found:
  Poke Portc, $80          ' Вперед
  Pause 3000
  Poke Portc, 0            ' Останов
  Stages = Stages + 1
  If Stages = 5 Then Finish
  Goto Scanning
Abandon:
  Poke Portc, 0            ' Останов
Flash:
  Poke Portc, $20
  Pause 500
  Poke Portc, $0
  Pause 500
  Goto Flash
Finish:
  Poke Portc, $30          ' Все светодиоды
  Pause 500
  Poke Portc, $0
  Pause 500
  Goto Finish

' Подпрограммы

ADread:
  Peek Adcon0, B0          ' Включение преобразователя
  Poke B0, 0
  Bit0 = 1
  Poke Adcon0, B0
  Poke Adcon1, $50         ' Тактовая частота равна 1/16 Fosc
  Pause 5
  Peek Adcon0, B0          ' Начало процесса преобразования
  Bit1 = 1
  Poke Adcon0, B0
Waiting:
  Pause 5
  Peek Adcon0, B0
  If Bit1 = 1 Then Waiting ' Проверка разряда GO/DONE
  Return                   ' Завершено
Scan:
  Poke Portc, $40          ' Назад
  Pause 1000
  Poke Portc, 0            ' Останов
  Pause 100                ' Достаточное время на остановку
```



**Листинг 6.10. Окончание**

```
  Poke Portc, $80          ' Вперед
  Pause 1000
  Poke Portc, 0            ' Останов
  Return
End
```

При разработке и тестировании этой программы полезно знать величины входных напряжений. В PICBASIC это просто. Короткая команда `Write 11, SampleB` помещает значение `SampleB` по адресу 11 в памяти EEPROM. Если микроконтроллер PIC установить в программатор, то это значение отобразится на экране компьютера. В подпрограмме отбора значений мы можем видеть десять последовательных значений `SampleA`, что дает нам представление о степени их разброса. По окончании разработки эти команды записи могут быть удалены.

### Обход препятствий

Структуру этой программы, в основном, совпадает с ассемблерной версией (листинг 6.11).



Соответствующий файл `Scoot04.txt` можно также найти на прилагаемом к книге компакт-диске в папке `Скутер\PICBASIC`.

**Листинг 6.11. Файл `Scoot04.txt`**

```
' Scoot04.txt

Symbol Status = $3
Symbol Porta = $5
Symbol Portb = $6
Symbol Portc = $7
Symbol Intcon = $B
Symbol Adresh = $1E
Symbol Adcon0 = $1F
Symbol Trisa = $85
Symbol Trisb = $86
Symbol Trisc = $87
Symbol Adcon1 = $9F
Symbol Ansel = $11E
Symbol Anselh = $11F

Symbol Reflect = B2
Symbol Backg = B3
Symbol Flag = BIT8

  Poke Intcon, 0     ' Запрет прерываний
```

**Листинг 6.11. Продолжение**

```
  Poke Porta, 0
  Poke Portb, 0
  Poke Portc, 0
  Poke Trisb, 0      ' Все разряды порта B - выходы
  Poke Trisc, 0      ' Все разряды порта C - выходы
  Poke Ansel, 3      ' Цифровой выход, за исключением RA0 и RA1
  Poke Anselh, 0
  Pause 5000

Scooting:
  Pause 100
  Flag = 0
  Poke Portc, $80    ' Двигатель вперед
  Pause 2000
  Poke Portc, 0      ' Останов двигателя
  Gosub Convert1
  Pause 100
  Write 4, B1
  If Flag = 1 Then Spin
  Goto Scooting

' Подпрограммы

Convert1:
  Flag = 0           ' Сброс флага отражения
  Poke Portc, $10    ' Включение передних светодиодов
  Pause 1000
  Gosub ADread
  Peek Adresh, Reflect      ' Считывание результата в Reflect
  Write 1, B2
  Poke Portc, 0      ' Выключение передних светодиодов
  Pause 1000
  Gosub ADready
  Peek Adresh, Backg        ' Считываем результат в Backg (фон)
  Write 2, B3
  Reflect = Reflect - Backg       ' Вычитание фонового света
  If Reflect < 0 Then Convert1    ' Неверный результат
  If Reflect < 25 Then Done       ' Отражение не обнаружено
  Write 3, B2
  Flag = 1           ' Если обнаружено отражение
Done:
  Return
Spin:
  High 6             ' Включаем зуммер
  Poke Portc, $50    ' Назад и поворот + D3
  Pause 5000
```



**Листинг 6.11. Окончание**

```
  Poke Portc, 0      ' Останов
  Low 6              ' Выключаем зуммер
  Return
ADread:
  Peek Adcon0, B0    ' Включение преобразователя
  Poke B0, 0
  Bit0 = 1
  Poke Adcon0, B0
  Poke Adcon1, $50   ' Тактовая частота 1/16 Fosc
  Pause 5
  Peek Adcon0, B0    ' Начало процесса преобразования
  Bit1 = 1
  Poke Adcon0, B0
Waiting:
  Pause 5
  Peek Adcon0, B0
  If Bit1 = 1 Then Waiting      ' Проверка разряда GO/DONE
  Return                        ' Завершено

End
```

# Глава 7. Робот “Андроид”

Вид робота “Андроид” традиционен для кинофильмов и видеоигр, однако его конструкция отличается новизной. Корпус робота сформирован из двух легких в обработке материалов малой плотности. Панели корпуса вырезаны из пенопласта. Это можно сделать с помощью острого ремесленного ножа и стальной линейки. Шарниры робота усилены бальзой. Конструкции скреплены ремесленным клеем. Полученный в результате корпус — очень жесткий и прочный, хотя для его сборки требуется совсем немного инструментов и навыков.

Придерживаясь модульного подхода, которому привержена данная книга, многие схемные блоки данного робота аналогичны блокам других роботов или только немного модифицированы.

Спецификация робота “Андроид”:

- работает от батареи 6 В;
- легкий корпус;
- три колеса: два задних приводных и одно поворачиваемое двигателем управляющее колесо впереди;
- микроконтроллер PIC 16F690;
- датчики: фоторезисторы, ввод по нажатию кнопки;
- исполнительные устройства: приводной двигатель, управляющий двигатель, двигатель манипулятора, зуммер, светодиоды, динамик.

Программы:

- игра «Ножницы, бумага, камень»;
- песни и танцы;
- поиск света;
- обход препятствий.

## Механика

Сборка робота начинается с шасси. Оно должно быть жестким и способным нести на себе конструкции двигателей, управляющей передачи и корпуса со всем его содержимым. Шасси следует изготавливать из фанеры, композитных листовых материалов или из нашего любимого материала: трехмиллиметрового листового пенополистирола. Габаритные размеры шасси с вырезами для колес показаны на рис. 7.1,



хотя конкретные размеры и форма зависят от используемых двигателей и колес (рис. 7.2).

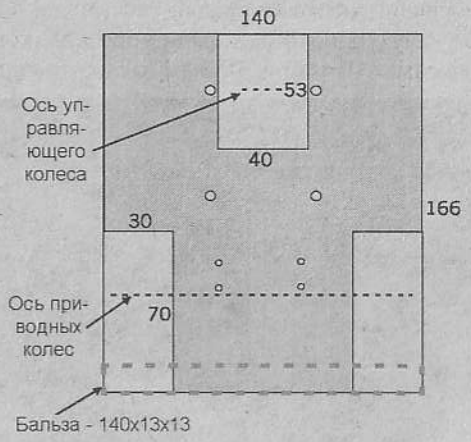



**Рис. 7.1.** Панель шасси с вырезами для колес

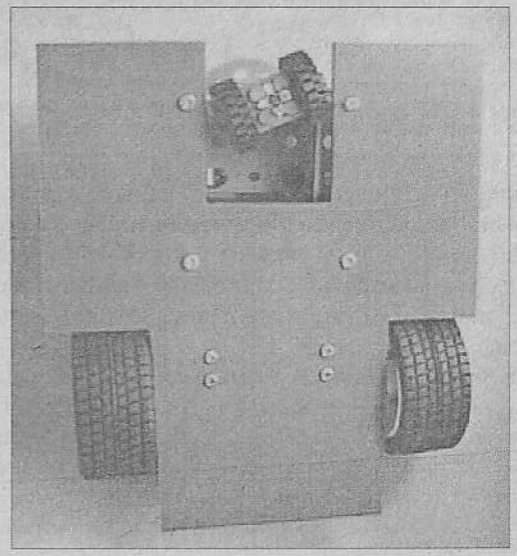


**Рис. 7.2.** Вид на шасси снизу показывает размещение колес в соответствующих прямоугольных областях. Когда корпус робота будет установлен, колеса не будут так хорошо видны. На этой фотографии также просматриваются сплющенные концы алюминиевой трубки (см. рис. 7.4)

Узел приводных колес состоит из двигателя на 6 В с коробкой передач и выходного вала с обеих сторон (рис. 7.3). Диаметр колес — около 56 мм. Мы использовали тот же колесный узел, что и в роботе "Скутер" (см. предыдущую главу), однако оси были обрезаны короче, чтобы расстояние между колесами составило 104 мм (от центра к центру).

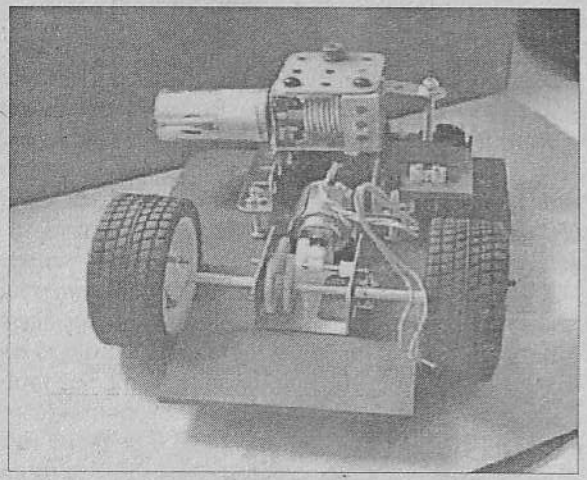


**Рис. 7.3.** Вид сзади на шасси с установленными колесами и двигателем. Панель шасси имеет большой выступ сзади. Позже к нему будет приклеена полоска бальзы для поддержки корпуса

Узел управляющего колеса собран из пары колес из конструктора Lego, смонтированных на специальном кубике, как и в роботе "Скутер" (рис. 7.4). На выходном валу двигателя с коробкой передач размещена червячная передача диаметром 4 мм, состыкованная с шестерней на 57 зубьев (рис. 7.5 и рис. 7.6). Альтернативный управляющий механизм обсуждается далее.

При монтаже шасси следует учитывать, как на следующем этапе сборки робота будет устанавливаться корпус, поддерживаемый шасси. Приводной и рулевой механизмы будут установлены в "ногах" корпуса. Корпус должен обеспечить им достаточно места с тем, чтобы они могли работать, не касаясь стенок. Кроме того, должен быть предусмотрен съем корпуса для внесения изменений или регулировки механизмов.

***Примечание***

Перед переходом к сборке корпуса проведите электронные и программные работы для приводного и рулевого двигателей. Легче сделать это сейчас и проверить их работу, когда открыт доступ к шасси и его механизмам.



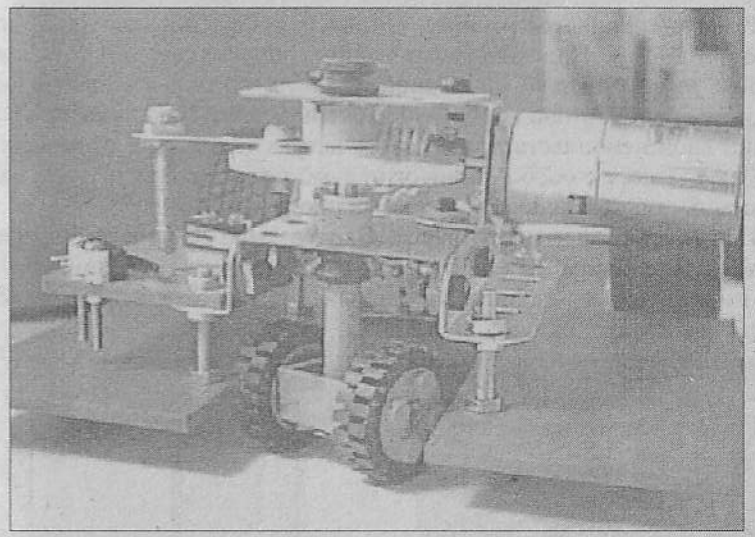


**Рис. 7.4.** Вид на шасси спереди демонстрирует управляющую коробку передач и узел управляющего колеса. Передаточная шестерня установлена на валу с треугольным сечением. На его нижний конец туго насажен отрезок алюминиевой трубки диаметром 5 мм. Центральное отверстие кубика рассверлено до диаметра 5 мм. Нижний конец трубки просаживается через это отверстие и прорезается на глубину около 6 мм для получения четырех "лепестков", которые затем загибаются для фиксации кубика на трубке

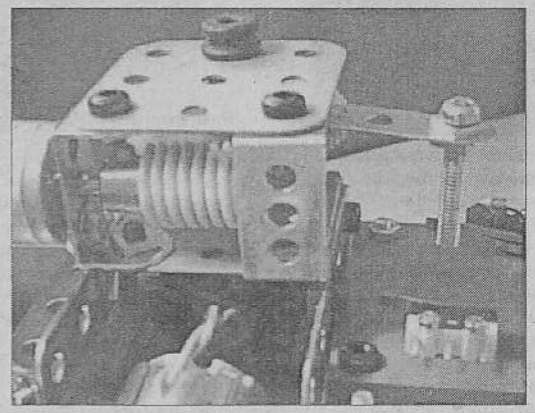


**Рис. 7.5.** Рычаг соединен с шестерней болтом, который выступает вниз, взаимодействуя с рычагами концевых выключателей. Они смонтированы на небольшой платформе из листового ПВХ, привинченной болтами к раме рулевого механизма. Эта фотография была сделана до того, как к выключателям были подсоединены провода

Корпус собирается их пенокартона и квадратного куска бальзы со стороной 13 мм. При конструировании корпуса можно фантазировать сколько угодно, однако мы решили сделать его максимально простым. При резке пенокартона используйте очень острый нож, поскольку внутренний слой пенопласта при резке зачастую отрывается от поверхностного слоя. Также с особой осторожностью режьте в близи углов панели. Во всем остальном пенокартон весьма прост в обработке.

Высота робота — дело вкуса. Следует только учитывать, что при малой высоте корпуса он не будет похож на андроида, а слишком большая высота повышает риск опрокидывания. Высота нашего опытного образца составила 360 мм (рис. 7.7).

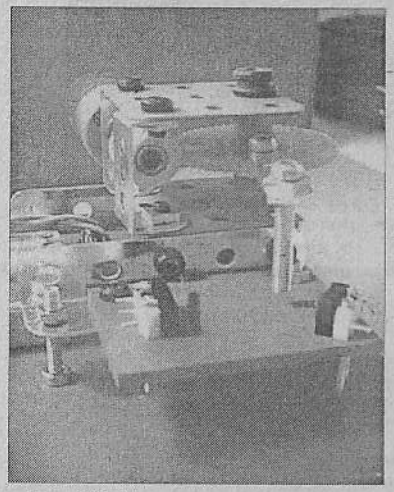


**Рис. 7.6.** На виде справа показана червячная передача и положение двух концевых выключателей. Управляющие колеса способны поворачиваться примерно на 15° в любую сторону от положения "прямо вперед". Концевые выключатели будут подключены проводами параллельно между входом микроконтроллера PIC (канал RA5) и линией 0 В. При замыкании любого из выключателей вход будет переходить в состояние низкого уровня

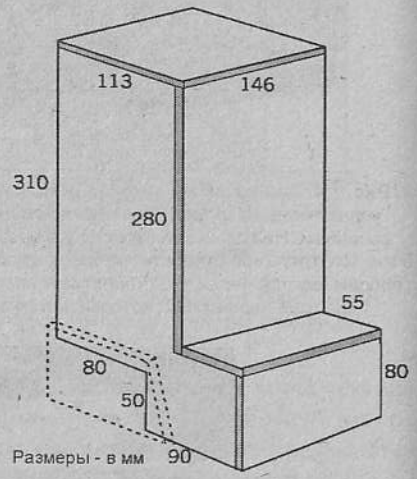



**Рис. 7.7.** Пенокартон — толщиной 5 мм. Здесь его обрезанные края выделены серым цветом. Боковые вырезы 80 × 50 мм предназначены для обеспечения дополнительного пространства для задних колес. Панель, обозначенная пунктиром, приклеивается на корпус, чтобы прикрыть эти вырезы

При планировании корпуса и расположения механизмов внутри него центр тяжести следует располагать максимально низко. Кроме того, он должен находиться над треугольником, образованным точками контакта трех колес с землей. Шасси с приводным и управляющим двигателями, а также — коробкой передач, необходимо располагать как можно ниже, поскольку это помогает опустить центр тяжести. Обеспечьте также максимально низкое размещение еще одного тяжелого узла: батареи.



На рис. 7.8 и рис. 7.9 показан один из способов сборки корпуса, однако читатель может использовать и собственный вариант.

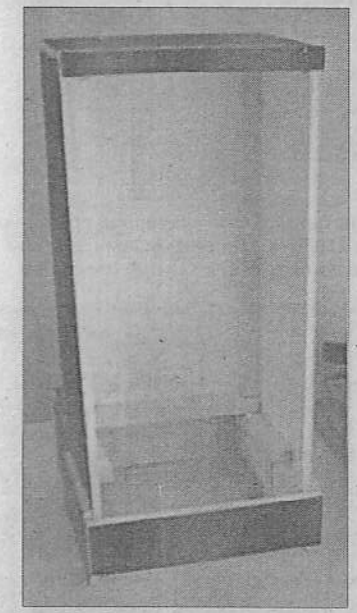


**Рис. 7.8.** Вид корпуса сзади. На нем видны рейки из бальзы, склеенные в углах. Две рейки внизу предназначены для поддержки полки, на которой монтируется батарея и основные платы. Для обеспечения свободного доступа эта полка к рейкам не приклеивается. Есть также две перекрестные рейки вверху и внизу. Длина крышки чуть больше расстояния между ними. Для ее установки крышка задвигается за верхнюю рейку, а затем пропускается вниз за нижней рейкой. Стопоры из бальзы не позволят ей упасть слишком низко

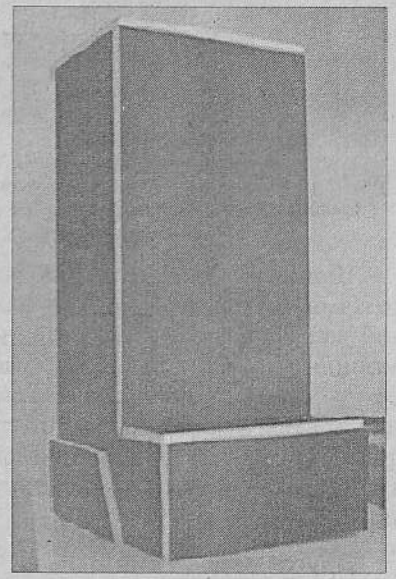


**Рис. 7.9.** Спереди робот "Андроид" темные панели с белыми торцами выглядят довольно неплохо. Следующий этап — создать руки робота

### Руки

Обе руки вырезаются из пенокартона (рис. 7.10). Если "Андроид" должен быть запрограммирован на игру "Ножницы, бумага, камень", то одна из рук должна быть насажена на вал шагового двигателя. В нашем опытном образце робот раскачивал правой рукой, чтобы показать свою активность. Другая его рука была приклеена, однако она также могла бы перемещаться собственным шаговым двигателем.

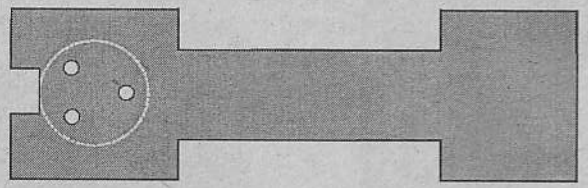


**Рис. 7.10.** Рука вырезается из прямоугольного куска пенокартона размерами примерно 130×40 мм. Подвижная рука имеет три отверстия в "плече" для ее фиксации болтами к шкивному колесу, которое крепит руку к валу

Шаговый двигатель закреплен болтами внутри корпуса. Его выходной вал проходит через отверстие в позиции "плеча". Рука зажимается между парой шкивов диаметром 28 мм и схватывается тремя болтами (рис. 7.11). Затем внутренний шкив плотно насаживается на вал двигателя.

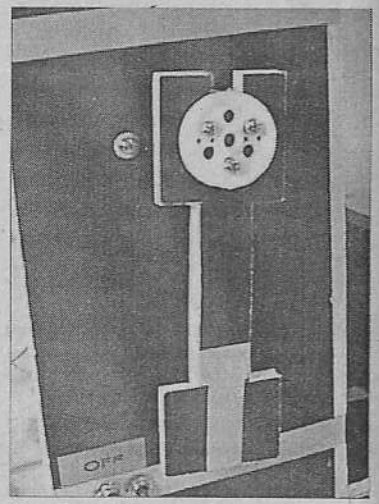


**Рис. 7.11.** Когда рука поворачивается в одну из фиксированных позиций, в вырезе на ее верхнем конце просматривается один из нескольких символов. Они могут обозначать ход робота в игре "Ножницы, бумага, камень". Существует несколько вариантов применения этой функции — особенно в игровых программах

### Другая периферия

Миниатюрный пьезоэлектрический динамик диаметром примерно 30 мм устанавливается внутри корпуса сразу под "головой" робота. Просверлите набор отверстий диаметром 2 мм в стенке корпуса для выпуска звука и приклейте позади них динамик ободом к корпусу. При необходимости прикройте перфорированную область снаружи, приклеив на нее круг из ткани.

Просверлите два отверстия в "голове" робота для вставки в них пары



светодиодов, выполняющих роль "глаз". В случае использования пенокартона отверстия могут быть чуть меньше по диаметру, чем диаметр светодиодов. Для светодиодов диаметром 5 мм отверстия диаметром 4,5 мм дадут плотную посадку.

| *Список приобретений для механической части робота* |
|---|
| Лист пенокартона толщиной 5 мм. |
| Лист ПВХ толщиной 3 мм или фанеры размерами около 140 x 166 мм. |
| Бальза — 13 мм x 13 мм x 2 м. |
| Ремесленный клей. |
| Двигатель постоянного тока и коробка передач с выходным валом на обеих сторонах. |
| Двигатель постоянного тока и коробка передач или шаговый двигатель для управления. |
| Шаговый двигатель постоянного тока для руки. |
| Детали червячного привода, если он используется. |
| Пара колес с шинами, приблизительно 60 мм в диаметре. |
| Металлическая трубка, которая входила бы в ступицу колес и надевалась на вал двигателя. |
| Шкивы (2 шт.) приблизительно 25 мм в диаметре для руки. |
| Разноцветная изоляционная лента для отделки корпуса робота. |
| Гайки и болты (главным образом М3). Болты — длиной 6 мм и 10 мм. |

## Электроника

Этот робот демонстрирует, как схемные модули, принадлежащие одному роботу, могут использоваться в другом. Их можно или сконструировать заново, или же разобрать старый робот и воспользоваться его платами. В случае с "Андроидом" мы решили выбрать второй вариант. Из ранее собранного робота "Скутер" мы сняли плату контроллера, плату управления двигателем и коммутационную плату, чтобы установить их в "Андроиде". Кроме того, мы воспользовались платой управления двигателем из робота "Искатель" (см. главу 9). Таким образом, все основные части "Андроида" были взяты из других роботов и потребовалось только собрать их воедино.

"Андроид" оснащен тремя двигателями: приводным, управляющим и двигателем руки. Последний необходим для перемещения правой руки робота, играющего в игру "Ножницы, бумага, камень". В нашей вер-

сии робота левая рука не двигается, хотя можно было бы установить двигатель и для нее.

Третий двигатель М3, который используется для управления, рассчитан на напряжение от 4,5 В до 18 В. При напряжении питания 6 В он работает медленно, но достаточно быстро для нашего случая. Напряжение не выходе Н-образного моста на 1,4 В меньше входного напряжения, поэтому двигатель получает только 4,6 В. Плата для его управления аналогична использованной в роботе "Скутер" (рассчитана на один двигатель). Другие два двигателя используют двойную плату управления из робота "Искатель".

Вся система работает от 6 В, если используется батарея из четырех щелочных элементов. Выключатель питания S1 установлен на правой стенке корпуса, ближе к "ногам" (убедитесь, что его не задевает вращающаяся рука).

### Плата контроллера

Эта плата аналогична плате контроллера робота "Скутер" (см. рис. 6.8), однако к ней добавлены три подтягивающих резистора номиналом 10 кОм, которые впаяны между точками E2–F2, E3–G3 и E4–H4.

Плата контроллера крепится болтами к полке справа, ближе к передней панели, где ее просто удалить и извлечь микроконтроллер PIC (рис. 7.12 и рис. 7.13).

Учитывая, что все ключевые схемные модули, за исключением одного, взяты готовыми из других роботов, казалось бы, единственное, что осталось, — это соединить их воедино. Однако здесь может возникнуть одна проблема. Н-образный мост управляющего двигателя не обеспечивает полное напряжение питания 6 В, в результате чего двигатель может вращаться слишком медленно или вообще не

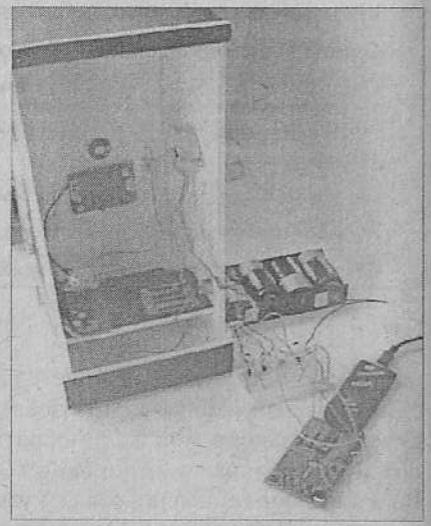


**Рис. 7.12.** Перед соединением плат были разработана и протестирована программа управления шаговым двигателем руки робота (установлен на правой стенке корпуса). На фотографии видны провода, ведущие к плате управления двигателем опытного образца. Макетная плата получает входные сигналы от микроконтроллера PIC, установленного в программатор PICkit 2



работать. Может потребоваться интерфейсная схема (рис. 7.14). Плата интерфейса (рис. 7.15) и двигатель управления при этом будут питаться от большего напряжения (скажем, 9 В), что потребует установки отдельного выключателя питания.

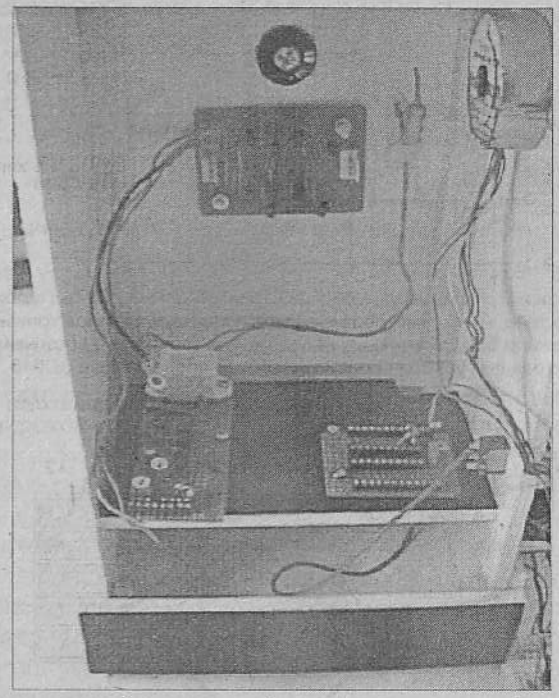


**Рис. 7.13.** Полку с платой контроллера (справа) и коммутационной платой (слева). В заднем правом углу полка срезана, чтобы можно было пропустить провода вниз к приводному и управляющему двигателю на шасси. Плата управления шаговым двигателем размещается на полке слева сзади. Батарея (4×AA) будет установлена по центру полки. Выключатель питания размещается на правой стенке, чуть выше платы контроллера. Плата управления приводным и управляющим двигателями установлена на задней стенке (на этом виде), а сразу же над ним размещен динамик. Провода от него проходят через кабельный держатель, на котором установлен конденсатор на 100 мкФ

Альтернативный вариант — подобрать двигатель (возможно, шаговый), который будет работать от напряжения 4,6 В. По сути, шаговый двигатель может оказаться предпочтительнее, поскольку он не получает питание от Н-образного моста, в силу чего может получить все 6 В от батареи.

Транзисторные ключи интерфейса представляют собой инверторы, поэтому сигнал от микроконтроллера при поступлении на двигатель ин-

вертируется. Если двигатель вращается не в том направлении, то просто поменяйте местами соединения E и F между интерфейсом и платой управления двигателем.

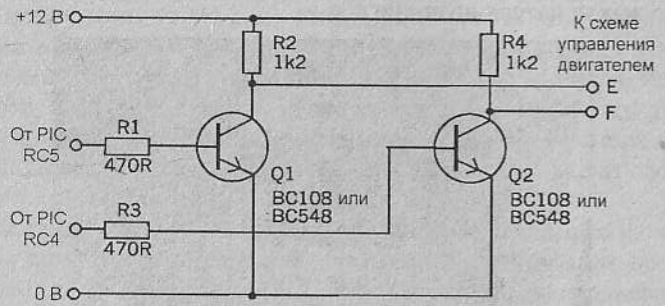



**Рис. 7.14.** Интерфейсная схема содержит два транзисторных ключа. Подойдут транзисторы почти любого типа. В опытном образце была использована пара транзисторов BC108, которые пролежали без применения в коробке для запчастей нескольких лет, однако с равным успехом подойдут и более современные BC548

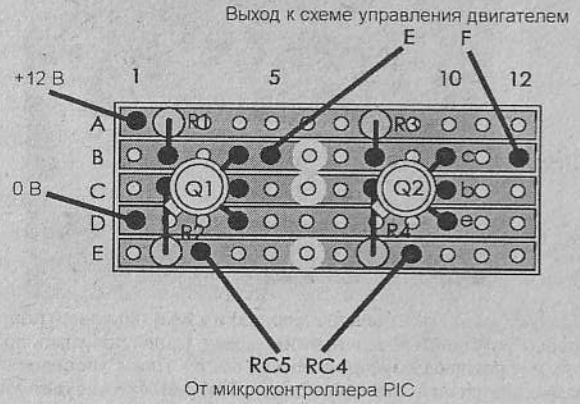



**Рис. 7.15.** На интерфейсной плате соединительные провода впаиваются прямо в отверстия. Их свободные концы оголены для вставки в гнезда на платах контроллера и управления двигателями. Напряжение питания снимается с дублирующих гнезд на этих платах

### Плата управления шаговым двигателем

Эта интерфейсная плата между микроконтроллером и входами шагового двигателя содержит четырех транзисторных ключа (рис. 7.16). Она питается от одного напряжения с контроллером (6 В), однако поддерживает и более высокое напряжение, если это необходимо для двигателя.



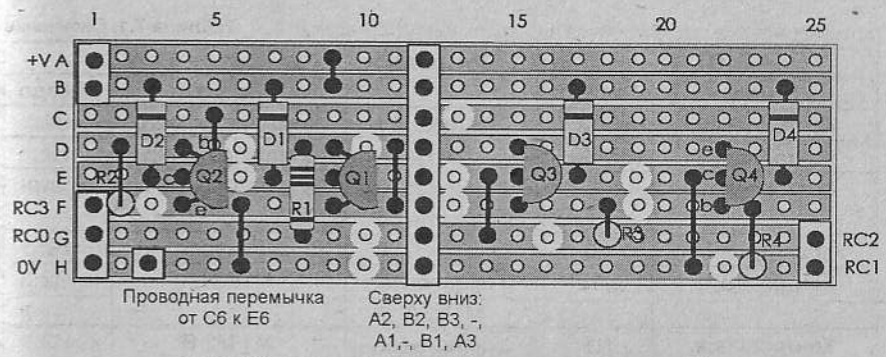



**Рис. 7.16.** Плата управления шаговым двигателем (M3)

Эту плату удобнее всего разместить сзади на полке, рядом с платой контроллера.

### Схема управления динамиком

Пьезоэлектрическим динамиком можно управлять непосредственно с выхода микроконтроллера PIC (см. рис. 3.35). Если звук недостаточно громкий, используйте транзисторный ключ для управления динамиком на 8 Ом. Схема содержит динамик в цепи коллектора и нуждается в защитном диоде (см. рис. 3.36).

При напряжении питания в 6 В и сопротивлении 8 Ом максимальный ток составляет 750 мА. Реализуйте ключ на небольшой плате, не включая резистор последовательно с динамиком и резистором базы на 470 Ом. Альтернативный (и более удобный) вариант — воспользоваться Q3 на коммутационной плате, а R5 заменить проводной перемычкой.

### Внеплатные соединения

В табл. 7.1 перечислены соединения для подачи питания и подключения двигателей. Проверяйте их в первую очередь.

**Таблица 7.1.** Соединения для подачи питания и подключения двигателей

| От | | Функция | К | |
|---|---|---|---|---|
| Плата | Гнездо | | Плата | Гнездо |
| Контроллера | C1 | Положительное питание | Коммутационная | A1 |
| Контроллера | D1 | | Управления M3 | A1 |
| Контроллера | E1 | | Выключатель питания S1* | |
| Управления M3 | B1 | | Управления M1/M2 | C2 |
| Выключатель питания S1 (общий)* | | | Положительный полюс батареи | |

**Таблица 7.1.** Окончание

| От | | Функция | К | |
|---|---|---|---|---|
| Плата | Гнездо | | Плата | Гнездо |
| Контроллера | C12 | Линия 0 В | Коммутационная | C1 |
| Контроллера | D12 | | Управления M3 | HI |
| Контроллера | E12 | | Отрицательный полюс батареи | |
| Управления M3 | H3 | | Управления M1/M2 | E2 |
| Контроллера | N12 | Управление M1/2, A | M1/M2 A | G2 |
| Контроллера | N1 | Управление M1/2, B | M1/M2 B | H2 |
| Контроллера | J1 | Управление M1/2, C | M1/M2 C | I2 |
| Контроллера | I1 | Управление M1/2, D | M1/M2 D | J2 |
| Управления двигателем | G20 | Выход двигателя A | Клемма двигателя M1* | |
| Управления двигателем | H2O | Выход двигателя B | Клемма двигателя M1* | |
| Управления двигателем | I20 | Выход двигателя C | Клемма двигателя M2* | |
| Управления двигателем | J20 | Выход двигателя D | Клемма двигателя M2* | |

Далее выполните соединения для шагового двигателя руки согласно табл. 7.2.

**Таблица 7.2.** Соединения для шагового двигателя руки робота

| От | | Функция | К | |
|---|---|---|---|---|
| Плата | Гнездо | | Плата | Гнездо |
| Контроллера | I12 | M3 | Управления M3, A1 | G1 |
| Контроллера | J12 | M3 | Управления M3, A3 | H25 |
| Контроллера | K12 | M3 | Управления M3, B1 | G25 |
| Контроллера | K1 | M3 | Управления M3, B3 | F1 |

Соединения с двигателем указаны согласно схеме платы с полосками, показанной на рис. 7.16, с применением восьмиконтактного разъема и гнезда.

Наконец, перечень соединений для других внеплатных компонентов (табл. 7.3).



**Таблица 7.3.** Соединения для других внеплатных компонентов

| От | | Функция | К | |
|---|---|---|---|---|
| Плата | Гнездо | | Плата | Гнездо |
| Контроллера | H1 | Кнопка | Клемма кнопки* | |
| Контроллера | G12 | LDRI | Коммутационная | F1 |
| Контроллера | F12 | Опорное напряжение компаратора 1 | Коммутационная | B1 |
| Контроллера | L12 | D1/D2 | Коммутационная | B22 |
| Контроллера | M12 | Динамик | Клемма динамика** | |
| Контроллера | H12 | Зуммер | Коммутационная | A22 |
| Контроллера | F1 | Рулевые концевые выключатели | Концевые выключатели | |

* Другой вывод кнопки и динамика подключается к 0 В. Аноды D1 и D2 подключены к положительному выводу источника питания.

** Динамик может включаться через Q2, установленный на коммутационной плате.

### Выводы микроконтроллера PIC

В табл. 7.4 перечислены подключения к выводам микроконтроллера PIC. Ксерокопия этой таблицы, прикрепленная к стенду, будет полезной при сборке и тестировании модулей системы.

**Таблица 7.4.** Подключения к выводам микроконтроллера PIC

| Порт | Вывод | Номер | Тип | Подключение | 0= | 1 = |
|---|---|---|---|---|---|---|
| A | RA0 | 19 | Аналог. вход | Опорное напряжение, компаратор 1 (C1IN+) | | |
| | RA1 | 18 | Аналог. вход | Передний фотоэлемент, компаратор 1 (C12IN–) | | |
| | RA2 | 17 | Вход | Выбор программы | | |
| | RA3 | 4 | Вход | Кнопочный переключатель | Нажат | Не нажат |
| | RA4 | 3 | Вход | Резерв (только ввод) | | |
| | RA5 | 2 | Вход | Рулевые концевые выключатели | Предел | Свободно |
| B | RB4 | 13 | Выход | Фары | Выкл. | Вкл. |
| | RB5 | 12 | Выход | Динамик | | |
| | RB6 | 11 | Выход | M1, приводной двигатель, A | Вперед | Назад |
| | RB7 | 10 | Выход | M1, приводной двигатель, B | Назад | Вперед |

Таблица 7.4. Окончание

| Порт | Вывод | Номер | Тип | Подключение | 0= | 1 = |
|---|---|---|---|---|---|---|
| C | RC0 | 16 | Выход | M3, двигатель руки, A1 | Вверх | Вниз |
| | RC1 | 15 | Выход | M3, двигатель руки, A3 | Вверх | Вниз |
| | RC2 | 14 | Выход | M3, двигатель руки, B1 | Вверх | Вниз |
| | RC3 | 7 | Выход | M3, двигатель руки, B3 | Вверх | Вниз |
| | RC4 | 6 | Выход | M2, двигатель управления, C | Влево | Вправо |
| | RC5 | 5 | Выход | M2, двигатель управления, D | Вправо | Влево |
| | RC6 | 8 | Выход | Зуммер | Выкл. | Вкл. |
| | RC7 | 9 | Выход | Резерв | | |

Строки табл. 7.4 сгруппированы по портам ввода-вывода. В ней перечислены все каналы, присутствующие в PIC16F690. В табл. 7.4 также показаны настройки линий A и B двигателей, обеспечивающих движение вперед/назад и останов. Для останова двигателя на обеих линиях должен присутствовать лог. 0 или лог. 1. Если мотор вращается не в том направлении, реверсируйте подключения к плате управления двигателем. То же самое относится и к линиям C и D двигателя M2.

Двигатель M3, двигающий руку робота, представляет собой шаговый двигатель с четырьмя линиями управления.

### Управление движением с помощью шагового двигателя

Обычный двигатель постоянного тока M2 заменен шаговым двигателем того же типа, что и для управления рукой робота. Его преимущество — в более высокой точности и скорости измерения направления. Он также может быть более компактным. Соответствующий перечень соединений представлен в табл. 7.5.

Таблица 7.5. Соединения в случае управления с помощью шагового двигателя

| От | | Функция | К | |
|---|---|---|---|---|
| Плата | Гнездо | | Плата | Гнездо |
| Контроллера | C1 | Положительное питание | Коммутационная | A1 |
| Контроллера | D1 | | Управление M3 | A1 |
| Контроллера | E1 | | Выключатель питания S1* | |
| Управления M3 | B1 | | Управления M2 | A1 |



Таблица 7.5. Окончание

| От | | Функция | К | |
|---|---|---|---|---|
| Плата | Гнездо | | Плата | Гнездо |
| Управления M2 | B1 | Положительное питание | Управления M1 | C1 |
| Выключатель питания S1 (общий) * | | | Положительный полюс батареи | |
| Контроллера | C12 | Линия 0 В | Коммутационная | C1 |
| Контроллера | D12 | | Управления M3 | H1 |
| Контроллера | E12 | | Отрицательный полюс батареи | |
| Управления M3 | H3 | | Управления M2 | H1 |
| Управления M2 | H3 | | Управления M1 | E2 |
| Контроллера | N12 | Управление M1 | M1 A | G2 |
| Контроллера | N1 | | M1 B | H2 |
| Контроллера | J1 | Управление M2 | M2 A1 | G1 |
| Контроллера | I1 | | M2 A3 | H25 |
| Контроллера | L1 | | M2 B1 | G25 |
| Контроллера | M1 | | M2 B3 | F1 |
| Контроллера | I12 | Управление M3 | M3 A1 | G1 |
| Контроллера | J12 | | M3 A3 | H25 |
| Контроллера | K12 | | M3 B1 | G25 |
| Контроллера | K1 | | M3 B3 | F1 |

Компоновка платы управления двигателем изменена на одну плату H-образного моста для приводного двигателя M1. Результирующая плата похожа на использованную в роботе "Скутер" (см. предыдущую главу). Новый управляющий двигатель, который мы обозначаем как M2, требует второй платы управления, наподобие использованной для M3.

### Выводы микроконтроллера PIC при управлении движением с помощью шагового двигателя

В табл. 7.6 перечислены подключения к выводам микроконтроллера PIC для случая управления движением робота с помощью шагового двигателя. Ксерокопия этой таблицы, прикрепленная к стенду, будет полезной при сборке и тестировании модулей системы.

Строки табл. 7.6 сгруппированы по портам ввода-вывода. В ней перечислены все каналы, присутствующие в PIC16F690. Здесь M2 — это шаговый двигатель управления направлением движения, а M3 — шаговый двигатель для перемещения руки робота. Оба эти двигателя управляются по четырем линиям.

## Программирование

Позаимствовав некоторые из плат робота "Скутер", "Андроид" может выполнять те же самые программы: "Hello World!", поиск света и обход препятствия. Распределение сигналов ввода-вывода для робота "Андроид" отличается от распределения в "Скутере", поэтому листинги должны быть отредактированы следующим образом:

- зуммер перемещен с RB6 (вывод 11) на RC6 (вывод 8);
- фары D1 и D2 перенесены с RC4 (вывод 6) на RB4 (вывод 11);
- боковой светодиод D3 на RC5 (вывод 5) в роботе "Андроид" не устанавливался, однако присутствует резервный канал RC7 (вывод 9), который можно для этого использовать;
- двигатель M1 перенесен с RC6/7 (выводы 8/9) на RB6/7 (выводы 10/11).

Основное различие заключается в том, что робот "Андроид" оснащен двигателем M2 для управления движением вместо эксцентрикового шкива. В тех местах листинга, где указывается, что робот изменяет направление движения путем небольшого отъезда назад, команда включения двигателя M1 в обратном направлении опускается. Вместо этого двигатель M2 программируется так, чтобы изменить направление движения.

### Прямолинейное движение

Первое, что необходимо сделать почти в любой программе, — повернуть управляющее колесо таким образом, чтобы робот двигался прямо вперед. Это колесо может остаться в угловой позиции после предыдущего этапа управления, поэтому, фактически, к началу нового этапа робот не знает, куда он ориентирован. Эта проблема отсутствует в роботах с управлением танкового типа.

Программа поворачивает управляющее колесо в одном направлении до тех пор, пока не сработает концевой выключатель. Теперь программа знает, куда повернуто колесо, и поворачивает его в обратном направлении на фиксированный угол, выводя в центральную позицию. Существует две версии этой программы: одна — для двигателя постоянного тока, а другая — для шагового двигателя.

Программа для двигателя постоянного тока при выводе управляющего колеса в центральную позицию полагается на тактирование, поэтому требует точной настройки и даже в этом случае может оказаться неточной. Программа для шагового двигателя очень точна при условии, что концевой выключатель точно позиционирован.



Невзирая на наличие концевого выключателя, управление с помощью двигателя постоянного тока в общем случае менее точное, чем управление с помощью шагового двигателя. Использование двигателя постоянного тока приводит к накоплению погрешности и необходимо использовать два конечных выключателя (левый и правый), чтобы предотвратить слишком большой угол поворота рулевого колеса в любом направлении. При использовании шагового двигателя достаточно только одного концевого выключателя для первичного центрирования колеса. После этого всякий раз при повороте колеса считаются шаги. Позиция колеса сохраняется как значение переменной, и робот всегда знает, куда он ориентирован.

Программа для управления с помощью двигателя постоянного тока показана в листинге 7.1, а соответствующий ей HEX-файл — в листинге 7.2.

Соответствующие файлы `And01.asm` и `And01.hex` можно также найти на прилагаемом к книге компакт-диске в папке `Андроид`.

**Листинг 7.1. Файл And01.asm**

```
;*****************************************************************
;                                                                *
;  Имя файла: and01.asm                                          *
;  Управление с помощью двигателя постоянного тока               *
;                                                                *
;*****************************************************************

  list        p=16F690    ;  Объявление процессора
  __CONFIG    0x30c4

; Банк0
  status      equ 03h
  porta       equ 05h
  portb       equ 06h
  portc       equ 07h
  intcon      equ 0bh
  pir1        equ 0ch
  pir2        equ 0dh
; Банк1
  trisa       equ 05h
  trisb       equ 06h
  trisc       equ 07h
; Банк2
  cm1con0     equ 19h
  ansel       equ 1eh
  anselh      equ 1fh
```

**Листинг 7.1. Продолжение**

```
; Метки
  delay0     equ 20h
  delay1     equ 21h
  delayn     equ 22h
  times      equ 23h

  goto start
  org 04h
  goto start

start
  bcf intcon, 7      ; Запрет прерываний
  bsf status, 5      ; Банк1
  clrf portb         ; Все выводы порта B - выходы
  clrf portc         ; Все выводы порта C - выходы
  bcf status, 5      ; Банк2
  bsf status, 6
  movlw 02h          ; Аналоговый вход RA0, RA1
  movwf ansel
  clrf anselh
  movlw 80h          ; Настройка компаратора 1. Опорное
n0                   ;  напряжение - на выводе 19, вход - 18
  bcf status, 6      ; Банк0
  bcf status, 5
  clrf porta
  clrf portb
  clrf portc

; Программа начинается здесь

  movlw 010h
  call longdelay
  call beeps
  call straight
  bcf portb, 6       ; Вперед
  bsf portb, 7
  movlw 0ah
  call longdelay
  bcf portb, 7       ; Стоп
  call beeps
  bsf portb, 6       ; Назад
  bcf portb, 7
  movlw 0ah
  call longdelay
  bcf portb, 6       ; Стоп
```



**Листинг 7.1. Продолжение**

```
flash
  bsf portb, 4       ; Включаем фары
  call delay
  bcf portb, 4       ; Выключаем фары
  call delay
  goto flash

; Подпрограммы

beeps
  movlw 05h
  movwf times
domore
  bsf portc, 6
  movlw 05h
  call longdelay
  bcf portc, 6
  movlw 05h
  call longdelay
  decfsz times, f
  goto domore
  return
straight
  btfss porta, 4           ; Левый микропереключатель замкнут?
  goto centre              ; Если замкнут
  bcf portc, 5             ; Поворачиваем колеса влево
  bsf portc, 4
  call delay
  bcf portc, 4             ; Перестаем поворачивать колесо
  goto straight            ; До полного поворота влево
centre
  bcf portc, 4
  bsf portc, 5
  movlw 030h               ; Это значение требует настройки
  call longdelay           ; Колесо вправо, к центру
  bcf portc, 5             ; Перестаем поворачивать
  return
delay
  decfsz delay0, f
  goto delay
  decfsz delay1, f
  goto delay
  return
longdelay
  movwf delayn
repeat
```

**Листинг 7.1. Окончание**

```
  call delay
  decfsz delayn, f
  goto repeat
  return

end
```

**Листинг 7.2. Файл And01.hex**

```
:020000040000FA
:020000000528D1
:0800080005288B138316860105
:1000100087018312031702309E009F0180309900F0
:1000200003138312850186018701103046202920A1
:1000300034200613861704A6208613292006171 7
:1000400086130A3046200613061641200612412068
:1000500024280530A3000717053046200713053074
:100060004620A30B2B280800051E3B2887120716E5
:1000700041200712342807128716303046208712 95
:100080000800A00B4128A10B41280800A200412034
:06009000A20B4728080046
:02400E00C430BC
:00000001FF
```

Программа для управления с помощью шагового двигателя показана в листинге 7.3, а соответствующий ей HEX-файл — в листинге 7.4.

Соответствующие файлы And02.asm и And02.hex можно также найти на прилагаемом к книге компакт-диске в папке Андроид.

**Листинг 7.3. Файл And02.asm**

```
;*****************************************************************
;
;  Имя файла: and02.asm
;  Управление с помощью шагового двигателя.
;
;*****************************************************************

  list       p=16F690
  __config   0x30c4

; Банк0
  pcl        equ 02h
  status     equ 03h
  porta      equ 05h
  portb      equ 06h
```



**Листинг 7.3. Продолжение**

```
  portc      equ 07h
  intcon     equ 0bh
; Банк1
  option_reg equ 01h
  trisa      equ 05h
  trisb      equ 06h
  trisc      equ 07h
  wpua       equ 15h
; Банк2
  cm1con0    equ 19h
  ansel      equ 1eh
  anselh     equ 1fh
; Метки
  delay0     equ 20h
  delay1     equ 21h
  delayn     equ 22h
  mask       equ 23h
  pointer    equ 24h
  count      equ 25h
  times      equ 26h

  goto start
  org 04h
  goto start

codes
  addwf pcl, f
  retlw 90h          ; Старший полубайт порта C
  retlw 50h
  retlw 60h
  retlw 0a0h
start
  bcf intcon, 7      ; Запрет прерываний
  bcf status, 5      ; Банк0
  bcf status, 6
  bsf status, 5      ; Банк1
  movlw 0fbh
  movwf trisa        ; Вывод 2 порта A - выход
  clrf trisb         ; Все выходы
  clrf trisc         ; Все выходы
  bcf option_reg, 7  ; Активизируем слабые подтягивающие резист.
  bsf wpua, 4        ; WPU на RA4
  bcf status, 5      ; Банк2
  bsf status, 6
  clrf ansel         ; Для цифрового ввода-вывода
  clrf anselh
```

**Листинг 7.3. Продолжение**

```
  bcf status, 6      ; Банк0
  bcf status, 5
  clrf porta
  clrf portb
  clrf portc

; Программа начинается здесь

  clrf pointer       ; Обнуляем pointer
  movlw 03h
  movwf mask
  movlw 010h
  call longdelay
  call beeps
  call straight      ; Для центрирования управляющего колеса
  bcf portb, 6       ; Вперед
  bsf portb, 7
  movlw 0ah
  call longdelay
  bcf portb, 7       ; Стоп
  call beeps
  bsf portb, 6       ; Назад
  bcf portb, 7
  movlw 0ah
  call longdelay
  bcf portb, 6       ; Стоп
flashem
  bsf portb, 4
  call delay
  bcf portb, 4
  call delay
  goto flashem

; Подпрограммы

delay
  decfsz delay0, f
  goto delay
  decfsz delay1, f
  goto delay
  return
longdelay
  movwf delayn
repeat
  call delay
  decfsz delayn, f
```



**Листинг 7.3. Продолжение**

```
  goto repeat
  return
beeps
  movlw 05h
  movwf times
domore
  bsf porta, 2
  movlw 05h
  call longdelay
  bcf porta, 2
  movlw 05h
  call longdelay
  decfsz times, f
  goto domore
  return
straight
  btfss porta, 4     ; Левый микропереключатель замкнут?
  goto centre        ; Если замкнут
  movlw 01h
  movwf count
  call cyclel        ; Поворачиваем колесо влево
  call delay
  goto straight      ; Пока не будет повернуто до предела влево
centre
  movlw 06h          ; Устанавливаем счетчик для 6 шагов (45гр.)
  movwf count
  call cycler
  return
cycler
  movf pointer, w
  decf pointer, f    ; Следующая позиция вправо
  andwf mask, w
  call codes
  movwf portc        ; Шаг двигателя
  call delay
  call delay
  decfsz count, f    ; Подсчитываем шаги
  goto cycler
  return
cyclel
  movf pointer, w
  incf pointer, f    ; Следующая позиция влево
  andwf mask, w
  call codes
  movwf portc
  call delay
```

**Листинг 7.3. Окончание**

```
  call delay
  decfsz count, f
  goto cycle1
  return

end
```

**Листинг 7.4. Файл And02.hex**

```
:020000040000FA
:020000000A28CC
:080008000A28820790345034ED
:100010006034A0348B13831203138316FB308500E6
:1000200086018701811315168312031 79E019F0114
:10003000031383128501860187 01A4010330A30005
:10004000103039203E204920061386170A30392007
:1005000086133E20061786130A303920061306162B
:10060003420061234202F28A00B3428A10B34286A
:100070000800A2003420A20B3A2808000530A60090
:10080000515053039200511053039 20A60B40280B
:100900008000 51E50280130A5005E2034204928A4
:1000A0000630A50054200800240 8A40323050520D9
:1000B000870034203420A50B542808002408A40A03
:1000C000230505208700342034 20A50B5E28080076
:02400E00C430BC
:00000001FF
```

Непосредственно перед меткой программы `start` расположена справочная таблица `codes`. Она содержит коды, которые выдаются на двигатель, чтобы он отработал один шаг влево или вправо. Переменная `pointer` указывает на текущий код, полученный двигателем. Следующий код, считываемый из таблицы, будет корректным кодом для продолжения поворота в данном направлении.

Кодов — всего четыре, поэтому указатель должен вернуться к коду 0 после указания кода 3 (или к коду 3 после кода 0 в случае обратного отсчета). Это легко обеспечивается с помощью маски, представляющей собой двоичное значение `00000011`. Оно подвергается логической операции "И" с указателем для получения двух младших разрядов. Фактическое значение указателя достигает 255 перед сбросом, однако это игнорируется за счет выполнения операции логического "И" с маской и младшими двумя разрядами указателя. Обратите внимание, что при использовании управляющего шагового двигателя зуммер переносится с вывода RC6 на RA2.



### Песни и танцы

Начнем с "песни". Подпрограмма для формирования прямоугольных тоновых сигналов была описана в главе 5. Она создает звуки заданной частоты и длительности, однако для этого ей необходимо дать три значения на обработку. Кроме того, данную подпрограмму необходимо отредактировать, чтобы звуковой сигнал поступал на заданный вывод.

В роботе "Андроид" динамик управляется по каналу RB5 порта B. Отредактируйте листинг для считывания `portb` вместо `portc`, `trisb` вместо `trisc` и `portb 5` вместо `portc 0`. Так, для получения звука "си" введите следующий список значений:

```
movlw 024h
movwf data1
movlw 020h
movwf datafinal
movlw 02ah
movwf lendata
call playit
```

Это часть листинга основной программы, поэтому введите после нее `call playit`. При желании выполнить этот фрагмент автономно в качестве демонстрации заблокируйте переход микроконтроллера PIC на подпрограммы, введя следующий цикл ожидания:

```
done goto done
```

Если необходимо, чтобы один и тот же тон звучал в программе несколько раз, то весь соответствующий код можно оформить в виде подпрограммы.

Пример программы воспроизведения роботом одной ноты можно найти на прилагаемом к книге компакт-диске в папке `Андроид` (файлы `And03.asm` и `And03.hex`).

Для получения последовательности, состоящей из двух–четырех нот, повторите листинг соответствующее количество раз, вызывая после каждого `playit`. Например, данная подпрограмма играет C, C', G':

```
movlw 024h          ; Воспроизводим C
movwf data1
movlw 020h
movwf datafinal
movlw 0ah
movwf lendata
call playit
movlw 010h          ; Воспроизводим C'
movwf data1
movlw 02Fh
```

```
movwf datafinal
movlw 015h
movwf lendata
call playit
movlw 0ah           ; Воспроизводим G'
movwf data1
movlw 029h
movwf datafinal
movlw 07eh
movwf lendata
call playit
```

При указанных настройках тоновые сигналы длятся в течении одной или большего числа секунд. Самый простой способ сделать их короче заключается в снижении значения в счетчике внешнего цикла с 5 до 3 или 2. Для правильной настройки более практично использовать справочную таблицу. Каждая позиция такой таблицы содержит три значения: `data1`, `datafinal` и `lendata`. Таблица для трехнотной "настройки" показана ниже. В ней одно за другим указаны три значения для каждой ноты.

Подпрограмма считывания таблицы выбирает три значения с помощью переменной `pointer`, помещает их в три переменные, а затем вызывает подпрограмму `playit`. Для следующей ноты `pointer` опять трижды инкрементируется, считывая и перенося данные всякий раз перед вызовом подпрограммы `playit`. Этот процесс продолжается до тех пор, пока не будут воспроизведены все ноты.

Листинг для трехнотной последовательности:

```
yourtune
  addwf pcl, f
  retlw 024h        ; Это data1, первая нота
  retlw 020h        ; Это datafinal, первая нота
  retlw 02ah        ; Это lendata, первая нота
  retlw 010h        ; Это data1, вторая нота
  retlw 02fh        ; Это datafinal, вторая нота
  retlw 054h        ; Это lendata, вторая нота
  retlw 0ah         ;  Это data1, третья нота
  retlw 029h        ; Это datafinal, третья нота
  retlw 07eh        ; Это lendata, третья нота
```

Подобную подпрограмму можно расширить на гораздо большее количество нот, вплоть до формирования завершенной мелодии. Вначале необходимо определить требуемые значения, используя метод, описанный в главе 5, а затем — сформировать таблицу.

Пример программы воспроизведения роботом мелодии можно найти на прилагаемом к книге компакт-диске в папке `Андроид` (файлы `And04.asm` и `And04.hex`).



А теперь переходим к танцам! Управление двигателями реализуется путем передачи последовательности кодов на приводной двигатель M1 и двигатель управления M2. Для управления предпочтительнее использовать шаговый двигатель, поскольку он может программироваться на быстрое вращение (однако не слишком быстрое, иначе он может потерять несколько шагов). Коды могут быть помещены в справочную таблицу в каждую четвертую (вперед/назад/стоп) и пятую (влево/вправо) позицию.

Коды "танца" запускают или останавливают приводной или управляющий двигатель, которые продолжают работать во время воспроизведения очередной ноты. Когда воспроизведение ноты завершено, микроконтроллер PIC переходит к справочной таблице, чтобы выяснить, что делать дальше. Двигатели могут быть реверсированы, остановлены или же режим их работы может быть оставлен без изменений. Робот может продолжать свое движение в течение воспроизведения нескольких нот. Хореография роботов — весьма увлекательное занятие.

Пример программы "танцев" робота можно найти на прилагаемом к книге компакт-диске в папке `Андроид` (файлы `And05.asm` и `And05.hex`).

### Ножницы, бумага, камень

Эта известная во всем мире стратегическая игра — одна из любимых у робота "Андроид". При умелом программировании его можно даже научить выигрывать.

"Ножницы, бумага, камень" — игра для двух игроков, которые стоят лицом друг к другу с правой рукой, спрятанной за спину. Состояние ладоней правых рук символизирует ножницы, бумагу или камень. Ножницам соответствуют вытянутые указательный и средний пальцы при поджатых остальных. Бумагу символизирует раскрытая ладонь, а камень — кулак.

По сигналу игроки резко вытягивают руки вперед, чтобы показать свой выбор: ножницы, бумагу или камень. Если оба выбрали один и тот же вариант, то результат ничейный, однако при выборе разных вариантов победитель определяется по следующим правилам:

- ножницы побеждают бумагу, потому что режут ее;
- бумага побеждает камень, потому может обернуть его;
- камень побеждает ножницы, потому что тупит их.

Такая простая игра может показаться тривиальной, однако некоторые люди воспринимают ее всерьез. Так, недавно (а именно, в мае 2005 года) в программе новостей BBC было сообщено, что известная компа-

ния по проведению аукционов Christie выиграла контракт на 10,5 миллионов фунтов стерлингов в результате победы в одном раунде этой игры против фирмы Sotheby.

Для того чтобы робот "Андроид" мог играть в эту игру, его правая рука должна управляться шаговым двигателем. Требуются два датчика: переключатель выбора программы и кнопка. Состояние игры индицируется "глазами" робота (светодиоды D1 и D2) и зуммером. В данном случае переключатель выбора программы служит, по сути, для выбора одной из двух стратегий игры.

Ход игры с роботом "Андроид" выглядит следующим образом:

1. Установите переключатель выбора программы в положение "Вкл." или "Выкл.". Положению "Выкл." соответствует хаотическая игра робота, победа в которой определяется волей случая. Для положения "Вкл." выполняется программа, в соответствии с которой робот учится играть на своем опыте.
2. Запустите программу. "Глаза" робота начнут светиться с половинной яркостью.
3. Когда будете готовы начать игру, нажмите и отпустите кнопку. "Глаза" робота потухнут, а затем мигнут один раз, когда кнопка будет отпущена.
4. Рука робота, вероятно, уже будет поднята. Нажмите и удерживайте кнопку. Рука повернется по часовой стрелке. Эта стадия привязана ко времени. Если кнопка не будет нажата сразу после этапа 3, то программа перейдет к шагу 5, однако рука при этом окажется в неверной позиции.
5. Когда рука будет направлена вертикально вниз, отпустите кнопку.
6. Робот делает свой выбор и включает зуммер. Когда зуммер выключится, робот поднимает руку, чтобы указать свой выбор. Вы делаете то же самое.
7. Выбор робота может быть прочитан в прорези на его плече или же определен визуально по углу порота руки: 45° означает ножницы, 90° — бумагу, а 135° — камень.
8. Робот не может распознать ваш выбор, поэтому ему необходимо сообщить, если он выиграл. В случае выигрыша "Андроида" нажмите кнопку один раз, если же результат ничейный, то не предпринимайте никаких действий. По истечении приблизительно одной минуты робот опустит свою руку, перейдя в состояние готовности к следующему раунду.



9. Для того чтобы сыграть еще раз при той же стратегии, нажмите кнопку для возврата к шагу 6. Стратегия может быть изменена, если предварительно изменить положение переключателя режима.

Программа — довольно длинная, однако примечательно то, что основная ее часть уделена подпрограмм управления вводом-выводом. Собственно игра, заложенная в подпрограмме `strategy`, — относительно короткая.

Программу можно как угодно расширять. Например, робот мог бы использовать динамик и генерировать различные звуковые сигналы на различных стадиях игры. Или же его можно запрограммировать на исполнение короткой песни и танца после каждой его победы, а также — на испускание какого-либо мрачного звука после поражения.

Следует отметить, что существует множество стратегий обучения, которые робот может принять на вооружение. Рассматриваемая программа структурирована таким образом, чтобы обеспечить возможность включения новых подпрограмм с заменой уже существующих. Попробуйте запрограммировать другую стратегию, которая, на ваш взгляд, может привести к победе.

В программе используются такие регистры общего назначения:

- `delay0`, `delay1` и `delayn` — для подпрограмм `delay` и `longdelay`;
- `ranval`, `bitn` и `bitm` — для генератора случайных чисел;
- `count` — содержит `01`, `10` или `11`, которые позже умножаются на восемь для получения количества шагов, на которые должен повернуться вал двигателя;
- `pointer` — указывает на коды в справочной таблице `code`;
- `mask` — служит для маскирования разрядов <7:2> регистра `randval`, выделяя разряды <1:0>;
- `waitime` — служит для установки таймаута в подпрограмме `ready`;
- `countcopy` — содержит копию первичного значения переменной `count` для использования в дальнейшем;
- `newcount` — содержит `01`, `10` или `11`, которые определяют дальнейшую реакцию робота. Если робот выигрывает, то значение `newcount` остается неизменным, если же проигрывает, то содержимое `newcount` изменяется с `01` на `10`, с `10` на `11` и с `11` на случайно выбранное число.

Блок-схема главной программы показана на рис. 7.17. Она начинается с запуска подпрограммы генерирования случайных чисел, которая будет выполняться до тех пор, пока вы не решите начать игру нажатием

кнопки. Поскольку временная привязка никогда не повторяется в точности, игра почти наверняка каждый раз будет начинаться с другого случайного числа.

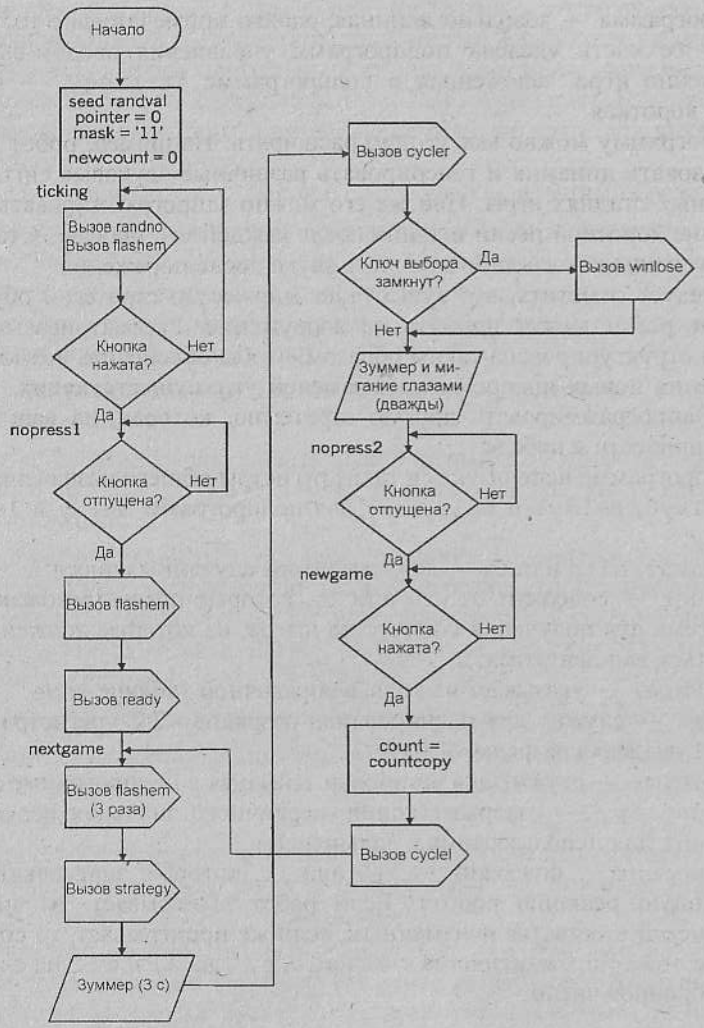



**Рис. 7.17.** Главная программа игры "Ножницы, бумага, камень"

Программа продолжается двойной проверкой состояния кнопки, чтобы микроконтроллер не начал работать с большой скоростью сразу



же после нажатия кнопки. Затем следует последовательность из четырех обращений к подпрограммам. "Глаза" робота мигают один раз, после чего выполняются подпрограммы `ready` и `moveit`, блок-схемы которых показаны на рис. 7.18. Они ориентируют руку робота вертикально вниз, если она еще не в этом положении.

До тех пор пока кнопка удерживается нажатой игроком, вызывается подпрограмма `moveit`, в которой подпрограмма `cyclel` продвигает шаговый двигатель на один шаг. Если кнопка отпущена, то подпрограмма `ready` обходит подпрограмму `moveit`. В любом случае ведется обратный отсчет значения переменной `waittime`, после которого позиция руки робота не может быть откорректирована.

При возврате в главную программу глаза робота мигнут три раза, чтобы показать, что "Андроид" готов к игре. Для того чтобы определиться с реакцией робота, вызывается подпрограмма `strategy`, блок-схема которой показана на рис. 7.19.

После повторного мигания глазами, подпрограмма считывает состояние ключа выбора программы. Если ключ разомкнут, то микроконтроллер PIC переходит к подпрограмме `strategyl`. В ней выбирается случайное число, и используется маска для выделения двух младших разрядов. Если они содержат `00`, то процесс повторяется до тех пор, пока состоянием разря-

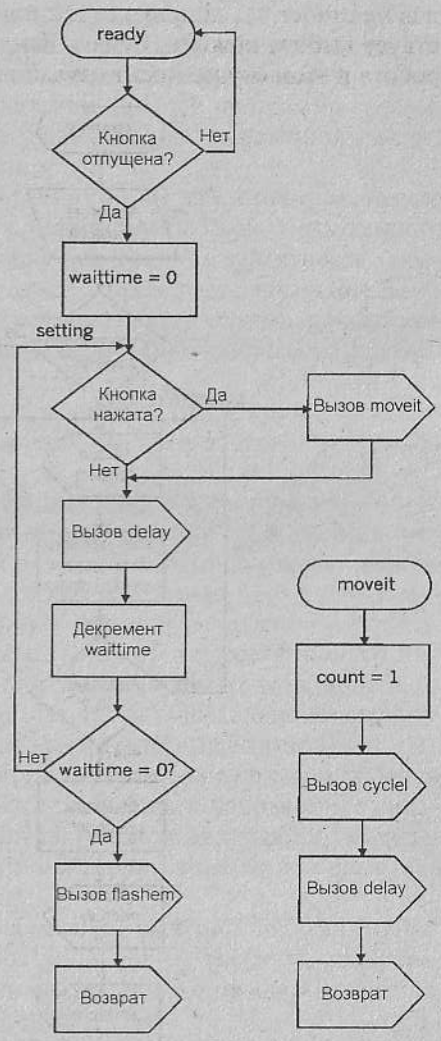



**Рис. 7.18.** Подпрограммы `ready` и `moveit`, которые готовят робота к игре

дов не станет 01, 10 или 11 (1, 2 или 3 в десятичном виде). Это соответствует выбору ножниц, бумаги или камня в игре, так что результат хода робота в этом случае носит случайный характер.

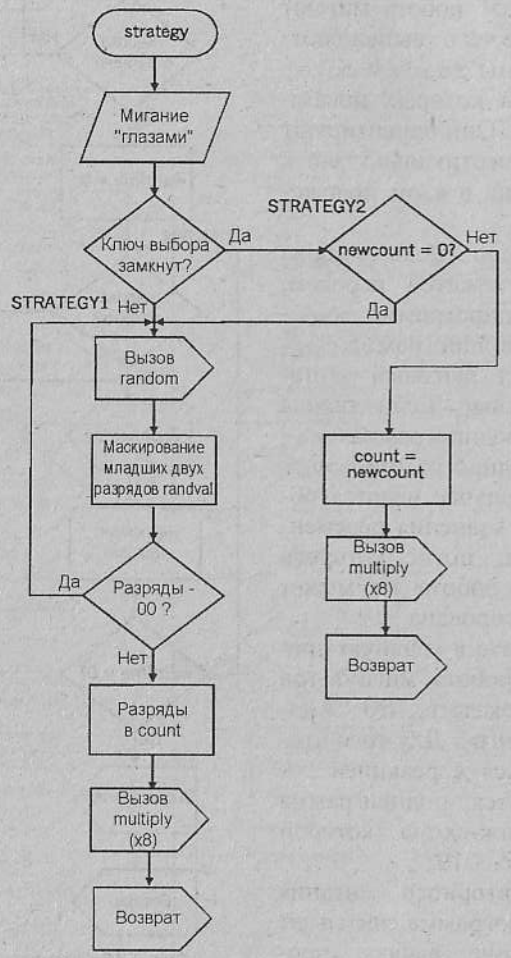



**Рис. 7.19.** Блок-схема подпрограммы `strategy`

Для преобразования результата в перемещение руки, значение (теперь в переменной `count`) умножается на восемь, чтобы дать 8, 16 или 24. Это достигается в результате трехкратного циклического сдвига влево переменной `count`. Восемь шагов двигателя поворачивают вал на



45°, поэтому значение в count приводит к повороту на 45°, 90° или 135°. Это значение сохраняется в переменной `countcopy` для использования в дальнейшем, когда рука робота возвращается в вертикальную позицию. Подпрограмма возвращает управление главной программе с разрядами (умноженными на восемь), сохраненными в переменной `count`, что определяет высоту поднятия руки.

В альтернативном режиме игры (`strategy2`) робот повторяет свою успешную игру до тех пор, пока она остается успешной. Цель данного алгоритма — дать роботу чуть больше, чем средняя вероятность победы. При входе в подпрограмму `strategy2` проверяется значение `newcount`. В первом раунде оно будет нулевым, в силу чего микроконтроллер PIC возвращается в подпрограмму `strategy1` для выбора случайного значения, которое заносится в `count`. Если это — не первая игра сеанса, то значение в `newcount` будет отличным от нуля. Оно присваивается переменной `count` и умножается на восемь перед возвратом в главную программу.

Значение в `newcount` может быть тем, которое привело робота к победе и повторяет его успешную игру. Однако, если в предыдущей игре робот проиграл, то переменной `newccount` обычно (однако не всегда) присваивается другое значение в подпрограмме `tryagain`. В результате в текущей игре робот попробует действовать иначе.

После возврата из подпрограммы `strategy` микроконтроллер PIC вызывает подпрограмму `cycler`, которая поворачивает руку робота на число шагов, содержащееся в `count`. На каждом шаге она обращается к справочной таблице `codes` и выдает код на шаговый двигатель.

В подпрограмме `cycler` указатель декрементируется на каждом шаге с тем, чтобы коды считывались из таблицы в направлении снизу вверх. В подпрограмме `cyclel`, используемой позже для автоматического опускания руки, указатель инкрементируется на каждом шаге, что дает считывание таблицы сверху вниз.

На этом игра завершается, и робот либо выиграл, либо проиграл. При использовании стратегии `strategy1` (ключ не замкнут), робот выдает звуковой сигнал и мигает глазами, после чего ожидает от игрока нажатия кнопки для запуска нового раунда.

В случае стратегии `strategy2` (ключ замкнут) робот переходит к подпрограмме `winlose`, в которой ожидает (метка `scoring`) нажатия игроком кнопки в случае победы робота. Длительность ожидания устанавливается значением в переменной `waittime`. Во время ожидания включен зуммер. Как только кнопка отпущена, `countcopy` копируется в `newcount`. Однако это значение в восемь раз превосходит исходное,

поэтому разряды 3 и 4 переменной countcopy копируются в разряды 0 и 1 переменной newcount. Это восстанавливает корректное значение newcount, приводя программу в состояние готовности к следующему раунду.

Затем микроконтроллер PIC ожидает (метка waiting) нажатия кнопки в течение обратного отсчета значения waittime до достижения им нуля. Если кнопка нажата, то PIC возвращается в главную программу со счетом побед, сохраненным в newcount.

Если кнопка нажата не будет, то программа переходит к метке result, где newcount инкрементируется, что дает роботу новый вид реакции. Однако, если предыдущее значение newcount было равно 11, то его инкрементирование приведет к выходу за рамки диапазона. В этом случае, подпрограмма tryagain выбирает случайное значение. Оно может быть тем же, что и предыдущее, однако есть шанс, что оно будет другим.

В любом режиме программа ожидает нажатия кнопки до того, как опустить руку робота, используя подпрограмму cyclel и декрементируя count. Программа переходит к nextgame, после чего робот трижды мигает "глазами", а зуммер сигнализирует о начале следующей игры.

Программа, реализующая рассмотренные выше алгоритмы, показана в листинге 7.5, а соответствующий ей HEX-файл — в листинге 7.6.

Соответствующие файлы And06.asm и And06.hex можно также найти на прилагаемом к книге компакт-диске в папке Андроид.

**Листинг 7.5. Файл And06.asm**

```
;*************************************************************
;
;  Имя файлы: and06.asm
;  Игра "Ножницы, бумага, камень"
;
;*************************************************************

  list      p=16F690
  __config  0x30c4

; Банк0
  pcl       equ 02h
  status    equ 03h
  porta     equ 05h
  portb     equ 06h
  portc     equ 07h
  intcon    equ 0bh
; Банк1
```



**Листинг 7.5. Продолжение**

```
  trisa     equ 05h
  trisb     equ 06h
  trisc     equ 07h
; Банк2
  ansel     equ 1eh
  anselh    equ 1fh
; Пункты назначения и флаг нулевого результата
  w         equ 00h
  f         equ 01h
  z         equ 02h
; Метки
  delay0    equ 20h
  delay1    equ 21h
  randval   equ 22h
  bitn      equ 23h
  bitm      equ 24h
  count     equ 25h
  pointer   equ 26h
  sps       equ 27h
  mask      equ 28h
  waittime  equ 29h
  countcopy equ 2ah
  delayn    equ 2bh
  newcount  equ 2ch

  goto start
  org 0004h1
  goto start

codes
  addwf pcl, f
  retlw 09h         ; Коды в младшем полубайте
  retlw 05h
  retlw 06h
  retlw 0ah
start
  bcf intcon, 7     ; Запрет прерываний
  bcf status, 5     ; Банк0
  bcf status, 6
  clrf porta
  clrf portb
  clrf portc
  bsf status, 5     ; Банк1
  clrf trisb        ; Все выходы
  clrf trisc        ; Все выходы
  bcf status, 5     ; Банк2
```

**Листинг 7.5. Продолжение**

```
  bsf status, 6
  clrf ansel        ; Для цифрового ввода-вывода
  clrf anselh
  bcf status, 6     ; Банк0
  bcf status, 5

; Программа начинается здесь

  bsf randval, 0    ; Генерирование случайных чисел
  clrf pointer      ; Очистка указателя
  movlw 03h
  movwf mask
  clrf newcount
ticking
  call flashem
  call randno
  btfsc porta, 3    ; Кнопка нажата?
  goto ticking
nopress1
  btfss porta, 3    ; Ожидание отпускания кнопки
  goto nopress1
  call flashem      ; Мигание глаз
  call ready        ; Для поворота руки вниз
nextgame
  call flashem
  call flashem
  call flashem
  call strategy     ; Возврат с установленным значением count
  bsf portc, 6      ; Включение зуммера на 3 с
  movlw 010h
  call longdelay
  bcf portc, 6      ; Выключение зуммера
  call cycler       ; Для индикации выбора
  btfss porta, 2    ; Если используется strategy2
  call winlose
  bsf portc, 6      ; Готовность к игре?
  call flashem
  call flashem
  bcf portc, 6
newgame
  btfss porta, 3
  goto newgame
nopress2
  btfsc porta, 3
  goto nopress2
  movf countcopy, w ; Рука вниз
```



**Листинг 7.5. Продолжение**

```
  movwf count
  call cycle1
  goto nextgame

; Подпрограммы

delay
  decfsz delay0, f
  goto delay
  decfsz delay1, f
  goto delay
  return
longdelay
  movwf delayn
repeat
  call delay
  decfsz delayn, f
  goto repeat
  return
randno
  clrf bitn
  clrf bitm
  btfsc randval, 5  ; Получение n
  bsf bitn, 0       ; Если n = 1
  btfsc randval, 6  ; Получение m
  bsf bitm, 0       ; Если m = 1
  movf bitn, w      ; n в w
  xorwf bitm, w     ; XOR m и n, результат в w
  addlw 0ffh        ; Установка флага переноса, если w = 1
  rlf randval,f     ; Новое случайное число в randval
  return
flashem
  bsf portb, 4
  call delay
  bcf portb, 4
  call delay
  return
ready
  btfss porta, 3    ; Ожидание отпускания кнопки
  goto ready
  movlw 040h
  movwf waittime
setting
  btfss porta, 3    ; Нажатие, пока рука идет вниз
  call moveit
  call delay
```

**Листинг 7.5. Продолжение**

```
  decfsz waittime, f
  goto setting
  call flashem
  return              ; Время истекло.
moveit
  movlw 01h
  movwf count
  call cyclel         ; На один шаг
  call delay
  return
cycler
  movf pointer, w
  decf pointer, f     ; Следующая позиция
  andwf mask, w
  call codes
  movwf portc         ; Шаг двигателя
  call delay
  decfsz count, f     ; Подсчет шагов
  goto cycler
  return
cyclel
  movf pointer, w
  incf pointer, f
  andwf mask, w
  call codes
  movwf portc
  call delay
  decfsz count, f
  goto cyclel
  return
strategy
  bsf portb, 4        ; Включение "глаз"
  movlw 05h
  call longdelay
  bcf portb, 4        ; Выключение "глаз"
  btfsc porta, 2
  goto strategy1
  goto strategy2
strategy1             ; Чисто случайно
  call randno
  movf mask, w
  andwf randval, w    ; Младшие 2 разряда
  btfsc status, z
  goto strategy1      ; Если 00
  movwf count
  call multiply       ; x 8
```



**Листинг 7.5. Продолжение**

```
  return
strategy2
  movf newcount, w
  btfsc status, z     ; Первый раз?
  goto strategy1      ; Да, используем случайное значение
  movwf count         ; Нет, count = newcount
  call multiply       ; x 8
  return
multiply
  bcf status, 0       ; Обнуление разряда переноса
  rlf count, f        ; count x 8
  rlf count, f
  rlf count, f
  movf count, w
  movwf countcopy     ; Запоминаем count x 8
  return
winlose
  movlw 030h
  movwf waittime
  bsf portc, 6        ; Включение зуммера. Робот победил?
scoring
  btfss porta, 3      ; Ожидание отпускания кнопки
  goto scoring
  bcf newcount, 0     ; Копирование countcopy (x 8)
  btfsc countcopy,    ; В newcount<1:0>
  bsf newcount, 0
  bcf newcount, 1
  btfsc countcopy, 4
  bsf newcount, 1
nopress3
  call delay
  decfsz waittime, f ; Время истекло
  goto waiting
  goto timeout
waiting
  btfsc porta, 3      ; Кнопка нажата = робот выиграл
  goto nopress3
  return              ; С выигрышным значением count в newcount
timeout
  bcf portc, 6        ; Время истекло! Выключаем зуммер
  goto result         ; Кнопка не нажата - робот проиграл
  return
result
  call rejectit       ; С проигрышным значением count в newcount
  bcf portc, 6        ; Проигрыш зарегистрирован, зуммер выключен
  return
```

**Листинг 7.5. Окончание**

```
rejectit
  incf newcount, f  ; Следующее значение count
  btfss newcount, 2 ; Если равно 4
  return
tryagain
  call randno       ; Для выбора нового случайного значения
  movf mask, w
  andwf randval, w
  btfsc status, z   ; Если 00
  goto tryagain
  movwf newcount    ; Следующий вид реакции в newcount
  return

end
```

**Листинг 7.6. Файл And06.hex**

```
:020000040000FA
:020000000A28CC
:080008000A28820709340534BF
:1000100006340A348B1383120313850186018701 8A
:1000200083168601870183120317 9E019F01031324
:1000300083122214A6010330A800AC0152204720ED
:1000400085191E28851D22285220572052205220 13
:10005000522079200717103042200713672005 1D12
:1000600095200717522052200713851D3528851922
:1000700037282A08A50070202628A00B3D28A10BB0
:100080003D280800AB003D20AB0B43280800A3012E
:10009000A401A21A2314221B241423082406FF3EC1
:1000A000A20D080006163D2006123D200800851D01
:1000B00057284030A900851D62203D20A90B5B28F0
:1000C000522008000130A50070203D2008002608BD
:1000D000A6032805052087003D20A50B67280800FA
:1000E0002608A60A2805052087003D20A50B7028B4
:1000F000080006160530422006120519802888 28B7
:10010000472028082205031980 28A5008E20080012
:10011000 2C0803198028A5008E2008000310A50DC7
:10012000A50DA50D2508AA0008003030A900071765
:10013000851D98282C10AA192C14AC102A1AAC145E
:100140003D20A90BA428A7288519A0280800 7137B
:10015000AA280800AD2007130800AC0A2C1D0800CF
:0E0160004720280822050319B028AC0008002B
:02400E00C430BC
:00000001FF
```

# Глава 8. Робот-игрушка

Некоторые люди не особо любят работать руками, или, возможно, они скучают от чрезмерного ручного труда и предпочитают программирование. Один из способов помочь таким людям — начать с готового корпуса, который можно приобрести в магазине для моделистов-робототехников. Другой способ — воспользоваться готовой игрушкой.

> ***Внимание!***
>
> Если игрушка новая или почти новая, и на нее еще действует гарантия изготовителя, то ценой вскрытия игрушки или вмешательства в нее любым другим способом может быть снятие гарантии. Лучше используйте старую игрушку (с разрешения ее владельца, конечно же).

В этой главе описано, как пластмассовой игрушечной машине были приданы свойства робота. Эта же идея применима к широкому диапазону игрушек. Наибольший простор для экспериментов дают игрушки на колесах. Впрочем, почему бы не использовать вместо машины, например, игрушечную утку?

Игрушки типа кукол и пушистых животных более проблематичны. Вероятно, оптимальный вариант в этом случае — робот на базе куклы-марионетки. Соберите механизм, дергающий за нитки, оборудуйте его несколькими датчиками, а затем запрограммируйте его таким образом, чтобы кукла ходила, танцевала или выполняла акробатические трюки. Попытайтесь придать роботу интеллект и способность обучаться.

Итак, если игрушечная машина вас не вдохновляет, поищите какую-нибудь другую игрушку, которую можно наделить робототехническими свойствами. Например, обдумайте вариант моторной лодки или даже парусника. Научите судно входить в крутой бейдевинд, поворачивать на другой галс и делать поворот оверштаг. Все, что ему при этом понадобится, — переключатель по наклону и датчик направления ветра.

Спецификация робота-игрушки:

- проект основан на недорогой или вышедшей из строя игрушке;
- рабочее напряжение зависит от установленных устройств;
- любое количество колес, включая их отсутствие;
- микроконтроллер PIC 16F690;

- датчики и исполнительные устройства — в соответствии с типом игрушки.

Программы:

- переменная скорость вращения вала двигателя;
- переключение звуковых эффектов;
- звуковой эффект выстрела.

## Механика

При выборе игрушки для этого проекта, убедитесь, что у нее внутри достаточно пространства для размещения батареи и нескольких плат. Если в ней уже есть батарейный отсек, — это очень хорошо. Приземистые машины для малышей обычно обеспечивает достаточно внутреннего пространства.

Также убедитесь в том, что пластмасса, из которой сделана игрушка, не раскрошится при сверлении. Конструкция должна допускать разборку и последующую сборку без повреждения. Игрушечная машина, использованная в рассматриваемом проекте, собирается с помощью винтов-саморезов, которые идеально подходят для разборки и повторной сборки корпуса.

Выбранная нами игрушечная машина приводится в действие толчком руки. Она не содержит приводного двигателя, и в ней нет никакого рулевого механизма. Ее основное достоинство — кнопка на боку, нажатие на которую приводит к воспроизведению звуковых эффектов и миганию светодиодов. Эта функция легко может быть автоматизирована, о чем будет сказано позже.

Основные механические задачи в данном случае — установка приводного и управляющего двигателей, а также, возможно, — шагового двигателя для выполнения других функций (например, подъема кузова самосвала). Большие игрушечные грузовики могут послужить основой для установки готового робототехнического манипулятора, придавая ему мобильность.

### Приводные колеса

Выбранная нами игрушечная машина имеет четыре колеса, которые вращаются в опорах по бокам корпуса. Мы решили, что в качестве приводных будут использованы задние колеса, а для управления — передние. Батарея состоит из двух щелочных элементов AA, дающие в сумме напряжение 3 В. Узел двигателя и коробки передач рассчитан на 3 В и имеет выходной вал с обеих сторон.



После рассмотрения различных вариантов было решено поместить колеса на концах выходных валов. Для этого на вал мы предварительно надели пластиковую трубку для обеспечения плотной посадки колес. Однако затем оказалось, что снять колеса с их оригинальных осей невозможно. В результате мы отказались от первоначальной идеи и заменили колеса игрушечной машины колесами от компании Tamiya, которые были установлены согласно рис. 6.3.

Приводной двигатель прикреплен болтами к днищу кузова таким образом, чтобы его валы свободно проходили через существующие отверстия в боковых опорах.

### Управление направлением движения

Управление реализовать сложнее, поскольку передние колеса не поворачиваются. Кроме того, они плотно окружены крыльями, в силу чего вокруг них мало места для поворотов после изменения конструкции. Решение — подъем передних колес с помощью управляющего узла, похожего (однако более простого) на тот, который использовался в роботе "Андроид".

Подобие заключается в применении пары колес из конструктора Lego с шинами (рис. 8.1).

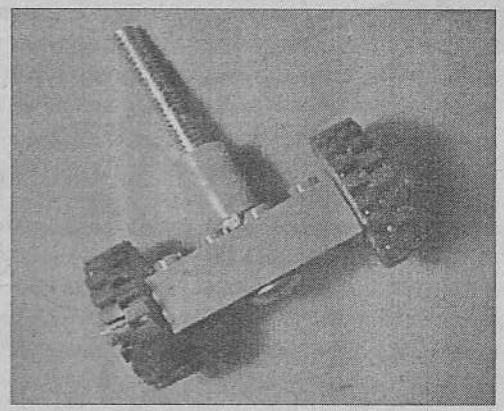


**Рис. 8.1.** Управляющий ролик. На болте установлены две пластмассовые втулки, задающие расстояние между колесами и днищем корпуса

В кубике из конструктора было просверлено отверстие диаметром 4,5 мм, через которое пропускается длинный болт длиной 50 мм и диаметром 3/16". Гайки, прилагаемые к таким болтам — квадратные, поэтому часть зубцов на кубике срезана (рис. 8.2), чтобы гайка поместилась между ними. Это предотвращает ослабление гайки.

Болт проходит вверх через отверстие диаметром 4,5 мм, которое просверлено в нижней части корпуса по центру и немного позади передних колес. Нижняя часть корпуса опирается на верхний конец втулки, высота которой обеспечивает зазор порядка 5 мм между передними колесами и землей. Такой зазор достаточно мал и незаметен.

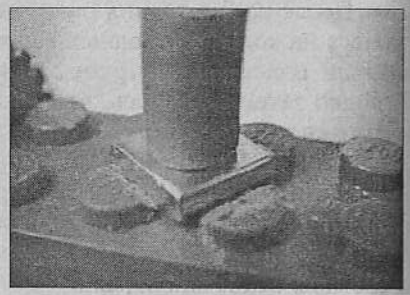


**Рис. 8.2.** Четыре зубца на кубике срезаются вокруг гайки так, чтобы она не могла поворачиваться

Верхний конец болта ролика идет вверх, проходя через отверстие в нижней части корпуса, под капот. На него надеты шайбы, зафиксированные гайками, поэтому узел поворачивается свободно, чрезмерно при этом не раскачиваясь. На верхнем конце болта между двумя гайками зажат пластмассовый шкив (рис. 8.3).

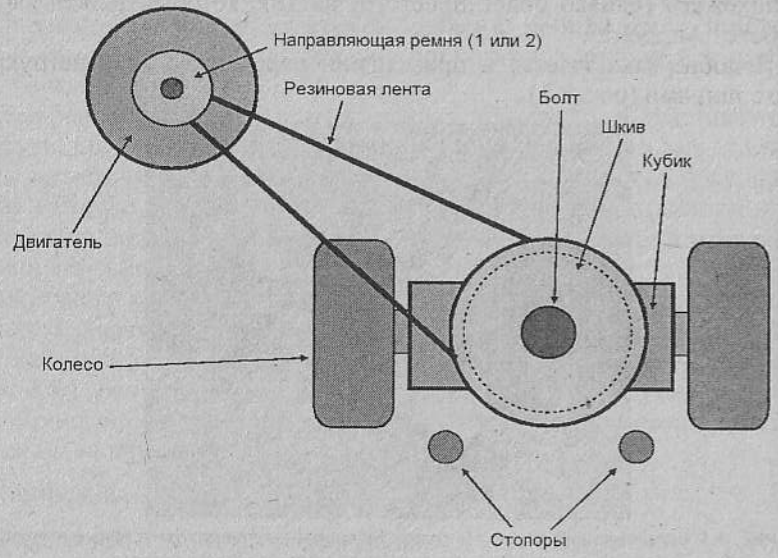



**Рис. 8.3.** Если посмотреть на рулевой механизм сверху вниз, то можно увидеть направленный вверх верхний конец болта ролика, на котором установлен шкив диаметром 25 мм, зафиксированный между двумя гайками. При этом поверх верхней гайки затянута третья гайка, выполняющая роль контргайки

Для управления используется двигатель постоянного тока на 3 В без коробки передач. Он невелик: всего лишь 20 мм в диаметре и 25 мм



в длину, — поэтому легко размещается в ограниченном пространстве внутри машины. У него нет никаких элементов для болтового крепления, поэтому он удерживается пружинным зажимом, привинченным болтами к боковой стенке корпуса. Этот зажим — того же типа, который используется для удержания батареи на 9 В типа PP3. Он жестко удерживает двигатель, однако позволяет смещаться ему вверх или вниз, чтобы "шкив" на выходном валу был на одном уровне со шкивом на валу ролика.

Собственно шкива на валу двигателя нет. Резиновый пассик вращается на самом валу. Он защищен от соскальзывания двумя направляющими, которые поставляются в комплекте шкивов от фирмы Tamiya.

Вниз с нижней стороны корпуса проходят два болта. Они установлены как стопоры, предотвращающие ролик от поворота больше, чем на 45° по обе стороны от положения "прямо вперед".

Резиновый пассик соединяет приводной вал и шкив. Когда короткий импульс поступает на двигатель, он вращается на высокой скорости, поворачивая звено шкив и ролик в том или ином направлении. Почти сразу дальнейший поворот блокируется одним из стопоров. Двигатель продолжает работать, однако пассик проскальзывает и шкив остается повернутым на 45° влево или вправо.

Управляющее действие в этом случае протекает гораздо быстрее, чем в других механизмах управления, и машина оживленно выписывает зигзаги. Такой робот хорош для программирования на объезд объектов или отслеживания траектории. Его можно даже обучить слалому между жестяными банками.

### Дальнейшая модернизация

В случае с игрушечной машиной забавным дополнением к датчикам будет ориентированный вниз фотоэлемент, установленный спереди на днище. Он может сопровождаться собственным источником света (предпочтительно инфракрасным) или же ориентироваться на окружающую освещенность. Когда машина будет ехать по поверхности стола, по достижении его края фотоэлемент зафиксирует изменение уровня освещенности. Запрограммируйте робот на немедленную реакцию в виде отъезда назад и поворота перед продолжением движения вперед.

Датчик, который обнаруживает магнитное поле Земли, необычен, однако недорог. Установленный на игрушечной машине (или в любом из других мобильных роботов), он обеспечивает ориентацию в пространстве. Его точность невелика, однако достаточно хороша для простых навигационных программ.

## Электроника

Звуковые эффекты были интересной функцией оригинальной игрушки, и мы решили сохранить их и в роботе. Они включались нажатием кнопки на боку машины, в робототехнической же версии они включаются автоматически с помощью реле.

Каждая кнопка содержит гибкий пластиковый колпачок, под которым находится маленький (4 мм в диаметре) диск из проводящей пены. Колпачок касается печатной платы в месте размещения двух полукруглых медных площадок с тонким зазором между ними (рис. 8.4). Когда колпачок прижат, проводящий диск перекрывает промежуток между контактными площадками, в результате чего включаются звуковые эффекты.

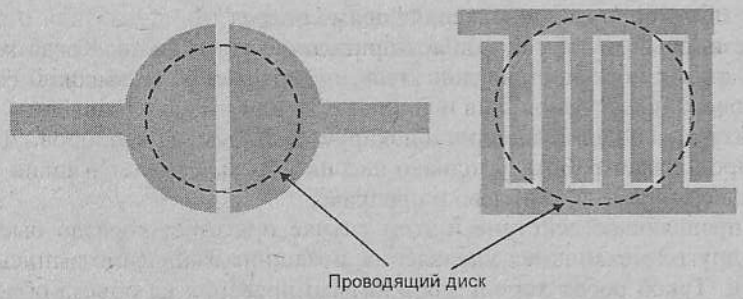



**Рис. 8.4.** Две формы медных контактных площадок, используемых в клавишах клавиатур

Переключение звуковых эффектов осуществляет микроконтроллер PIC через реле, включенное параллельно с кнопкой. Для подключения реле к плате звуковых эффектов игрушки аккуратно удалите колпачок кнопки с печатной платы. При включенном напряжении питания попробуйте кратковременно закоротить две контактные площадки. Это должно активизировать звуковой эффект. Если это так, то припаяйте провод к каждой контактной площадке и подсоедините их к выводам реле. Когда выходной сигнал микроконтроллера PIC запитает реле, его контакты замкнутся, соединив контактные площадки кнопки и тем самым включив звук.

Звуки добавляют к действиям робота реализма. При наличии еще одной подобной игрушки с другим набором эффектов ее плату звуковых эффектов можно установить вместо первой. Однако, прежде, чем сделать это, убедитесь, что рабочие напряжения обеих плат совпадают.

Электронная система включает в себя плату микроконтроллера, а также платные и внеплатные схемы для других установленных устройств. Как приводной, так и управляющий двигатель должен быть реверсивными, поэтому для каждого двигателя требуется транзисторный H-образный мост или плата реле. Просмотрите эту книгу, чтобы подыскать другие функции, которые можно было бы ввести в игрушку-робота.



Плата звуковых эффектов работает от напряжения 3 В от встроенного батарейного отсека. Второй отсек содержит еще два элемента AA (или AAA, если недостаточно места) и включен последовательно со встроенным. Две батареи в сумме дают 6 В для питания платы контроллера, транзисторного ключа включения реле и H-образных мостов, которые управляют управляющим и приводным двигателями.

## Программирование

### Рулевое управление

Для полного поворота управляющих колес до упора в каком-либо направлении достаточно импульса продолжительностью, соответствующей одному обращению к подпрограмме `delay`. Для того чтобы выставить управляющие колеса в положение "прямо вперед", вначале посылается импульс, поворачивающий колеса полностью вправо или влево, а затем — более короткий импульс, смещающий колеса в противоположном направлении, выставляя их в среднее положение. Этот импульс получают в результате вызова подпрограммы `shortdelay` со значением `040h` в качестве начального значения переменной `delay1`. Откорректируйте это значение в соответствии с используемым двигателем. Два импульса занимают долю секунды, поэтому получится эффект плавного изменения направления.

### Управление скоростью вращения вала

Эта подпрограмма находит различные области применения. Идея заключается в том, что питание подается на двигатель как последовательность импульсов с амплитудой, соответствующей полному напряжению питания (напряжению, снимаемому с H-образного моста). Однако соотношение между длиной импульсов и интервалами между ними можно изменять программно. Чем короче импульсы (и, следовательно, чем больше интервалы между ними), тем меньше энергии подается на двигатель и тем медленнее вращается его вал. Этот способ управления скоростью предпочтительнее варьирования напряжения, поскольку снижает вероятность останова двигателя или его отказа при пуске на малых оборотах.

Частота следования импульсов составляет около 100 Гц, так что двигатель будет работать плавно. Эта же схема может использоваться для снижения яркости свечения лампочки или светодиода.

Главная программа — короткая (рис. 8.5). Она устанавливает значения трех основных параметров последовательности прямоугольных импульсов, управляющей двигателем. В данной демонстрационной версии они вводятся в программу перед ее компиляцией и запуском. В реальном приложении они вычислялись бы программно.

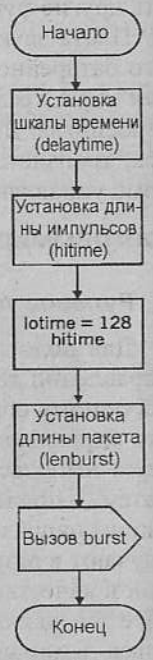



**Рис. 8.5.** Программа регулирования скорости вращения вала двигателя

Один период следования импульсов разделяется на 128 (08h) небольших временных блоков. Длина одного блока сохраняется в переменной `delaytime`, которая может быть установлена на любое значение между 1 и 256 (ffh). Вначале выходной сигнал переводится в состояние высокого уровня, которое удерживается на протяжении числа блоков, заданных переменной `hitime`. Затем выходной сигнал переводится в состояние низкого уровня, которое удерживается на протяжении числа блоков, заданных переменной `lotime`. Значение `hitime` вводится вручную, а значение `lotime` вычисляется в результате вычитания `hitime` из 128. Выход удерживается в состоянии низкого уровня в течение времени `lotime`.

Один период следования прямоугольных импульсов повторяется определенное число раз (`lenburst`) для формирования пакета выходных импульсов, управляющий двигателем в течение заданного отрезка времени. Программа могла бы быть адаптирована для питания двигателя в течение более длительного времени или же до того момента, пока не наступит определенное событие.

Период следования прямоугольных импульсов — постоянный (`hitime + lotime`), однако соотношение `hitime:lotime` может изменяться в диапазоне от 128:0 (двигатель работает на полной скорости) до 0:128 (двигатель выключен). На практике двигатель требует минимального значения `hitime` для поддержания небольшой частоты оборотов, поэтому полный диапазон для коэффициента не используется. Аналогично — и в случае управления светодиодами.



Программа управления двигателем по каналу RC0 с применением транзисторного ключа показана в листинге 8.1, а соответствующий ей HEX-файл — в листинге 8.2.

Соответствующие файлы `Toy01.asm` и `Toy01.hex` можно также найти на прилагаемом к книге компакт-диске в папке `Игрушка`.

**Листинг 8.1. Файл `Toy01.asm`**

```
;*****************************************************************
;                                                                *
;  Имя файла: toy01.asm                                          *
;  Управление скоростью вращения вала.                           *
;                                                                *
;*****************************************************************

  list       p=16F690            ; Определение процессора
  __CONFIG   0x30c4

; Банк0
  status     equ 03h
  portb      equ 06h
  portc      equ 07h
; Банк1
  trisb      equ 06h
  trisc      equ 07h
; Банк2
  ansel      equ 1eh
  anselh     equ 1fh
; Метки
  lenburst   equ 23h
  hitime     equ 24h
  lotime     equ 25h
  hicount    equ 26h
  locount    equ 27h
  delayx     equ 28h
  lencount   equ 29h
  delaytime  equ 2ah

  org 00h
  goto start
  org 04h
  goto start

start
  bsf status, 5      ; Банк1
  clrf trisc         ; Порт C - все выходы
  bcf status, 5      ; Банк2
```

**Листинг 8.1. Продолжение**

```
  bsf status, 6
  clrf ansel        ; Цифровой ввод-вывод
  clrf anselh       ; Цифровой ввод-вывод
  bcf status, 6     ; Банк0
  bcf status, 5

; Программа начинается здесь

  clrf portc
  movlw 010h        ; Длительность одного временного блока
  movwf delaytime
  movlw 70h         ; Число блоков для сигнала высокого уровня
  movwf hitime
  sublw 080h        ; Число блоков для сигнала низкого уровня
  movwf lotime
  movlw 050h        ; Длительность пакета
  movwf lenburst
  call burst        ; Формирование пакета импульсов
  bcf portc, 0
  bsf portc, 3      ; Светодиод на RC3 индицирует о 2-м пакете
  movlw 02h         ; Параметры для 2-го пакета
  movwf hitime
  sublw 080h
  movwf lotime
  movlw 0C0h
  movwf lenburst
  call burst
  bcf portc, 3
  endit goto endit

; Подпрограммы

burst
  movf lenburst, w  ; lencount = lenburst
  movwf lencount
speed
  movf hitime, w    ; hicount = hitime
  movwf hicount
  movf lotime, w    ; locount = lotime
  movwf locount
  bsf portc, 0      ; Включение выхода (высокий уровень)
hiphase
  call delayit      ; Сохраняет высокий уровень, пока hicount
  decfsz hicount, f ;  не уменьшится до нуля
  goto hiphase
  bcf portc, 0      ; Выключение выхода (низкий уровень)
```



**Листинг 8.1. Окончание**

```
lophase
  call delayit      ; Сохраняется низкий уровень, пока locount
  decfsz locount, f ;  не уменьшится до нуля
  goto lophase
  decfsz lencount, f       ; Пакет продолжается, пока
  goto speed               ; lencount не уменьшится до нуля
  return
delayit
  movf delaytime, w        ; Формирование задержки, пока
  movwf delayx             ;  delayx (= delaytime)
nextloop                   ;  не уменьшится до нуля
  decfsz delayx, f
  goto nextloop
  return

end
```

**Листинг 8.1. Файл Toy01.hex**

```
:020000040000FA
:020000000528D1
:0800080005288316870183120D
:1000100003179E019F0103138312870110030AA006A
:100020007030A400803CA5005030A30022200710AF
:1000300087150230A400803CA500C030A300222018
:10004000871121282308A9002408A6002508A70055
:100500007143320A60B292807103320A70B2D28BF
:10060000A90B242808002A08A800A80B3528080096
:02400E00C430BC
:00000001FF
```

Блок-схема подпрограммы `burst` показана на рис. 8.6. Эта подпрограмма может использоваться для увеличения реализма поведения мобильного робота. Вместо того, чтобы рывком срываться с места и резко останавливаться, робот будет плавно ускоряться, начиная движение, и изящно тормозить при остановках. Для этого подпрограмма `burst` вызывается с фиксированными значениями переменных `delaytime` и `lenburst` и исходным значением переменной `hitime`. Затем она входит в цикл, в котором `hitime` постепенно инкрементируется или декрементируется.

### Звуковая имитация пулеметной очереди

Эта программа формирует пакет сигналов белого шума, который создает звук, напоминающий пулеметную очередь. Это — удачное до-

полнение к роботу-игрушке военного типа. Звук можно настраиваться, так что данная программа может иметь и другие сферы применения.

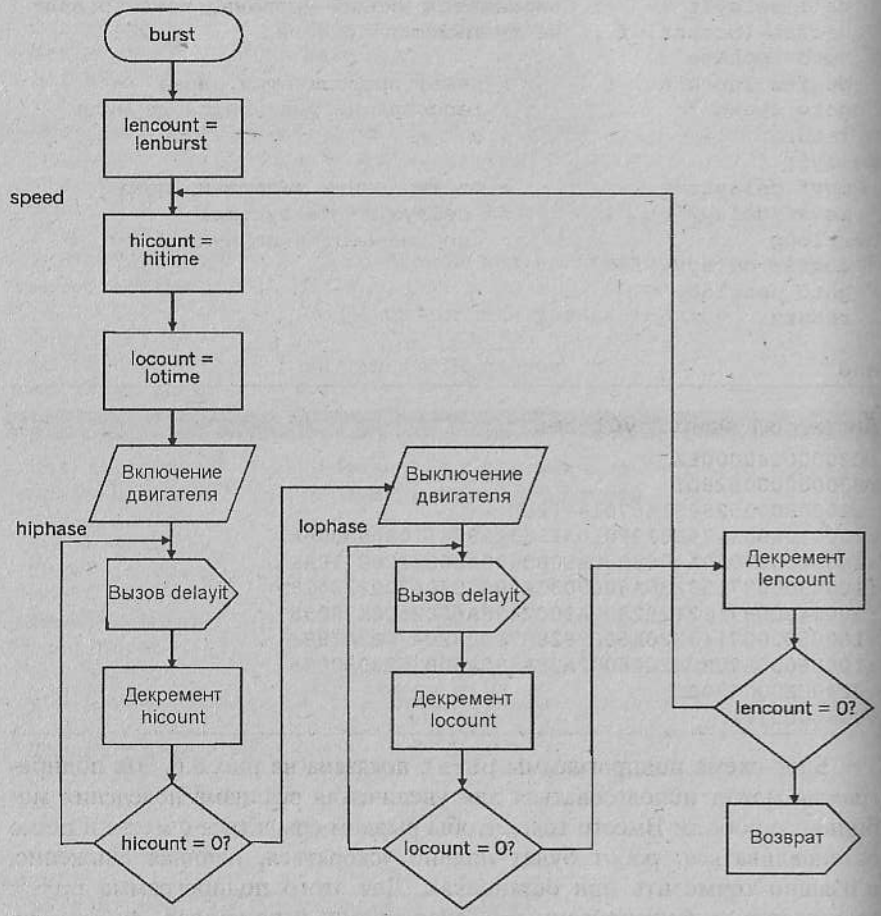



**Рис. 8.6.** Подпрограмма `burst` формирует последовательность импульсов фиксированной частоты и длительности, однако — с переменным коэффициентом заполнения

Генератор случайных чисел работает в цикле, с большой скоростью формируя последовательность значений переменной `randval`. После формирования каждого нового числа считывается (с помощью операции логического "И" `randval` с 080h) его разряд <7>, и канал RC7 порта C устанавливается в то же значение.



В результате на выходе RC7 выводится случайная последовательность значений. Она поступает на транзисторный ключ, в цепь коллектора которого включен динамик. Случайные уровни сигнала формируют пакет сигналов белого шума.

Программа имитации пулеметной очереди показана в листинге 8.3, а соответствующий ей HEX-файл — в листинге 8.4.

Соответствующие файлы `Toy02.asm` и `Toy02.hex` можно также найти на прилагаемом к книге компакт-диске в папке `Игрушка`.

**Листинг 8.3. Файл `Toy02.asm`**

```
;*****************************************************************
;
;  Имя файла: Toy02.asm
;  Формирование белого шума.
;
;*****************************************************************

  list      p=16F690
  __config  0x30c4

  status    equ 03h
  portc     equ 07h
  intcon    equ 0bh
  trisc     equ 07h
  ansel     equ 1eh
  anselh    equ 1fh
  w         equ 00h
  f         equ 01h
  delay0    equ 20h
  delay1    equ 21h
  randval   equ 22h
  bitn      equ 23h
  bitm      equ 24h
  count     equ 25h

  goto start
  org 0004h
  goto start

start
  bcf intcon, 7     ; Запрет прерываний
  bcf status, 5     ; Банк0
  bcf status, 6
  clrf portc
  bsf status, 5     ; Банк1
```

**Листинг 8.3. Продолжение**

```
  clrf trisc        ; Все выходы
  bcf status, 5     ; Банк2
  bsf status, 6
  clrf ansel        ; Для цифрового ввода-вывода
  clrf anselh
  bcf status, 6     ; Банк0
  bcf status, 5

; Программа начинается здесь

  bsf randval, 0    ; Формирование случайных чисел
  movlw 050h        ; Длина пакета
  movwf count
repeat
  call randno
  movlw 080h
  andwf randval, w  ; Получаем разряд <7>
  movwf portc       ; Передаем его в RC7
  call shortdelay
  decfsz count, f
  goto repeat
  bcf portc, 7      ; В случае, если последнее значение = 1
  endit goto endit

; Подпрограммы

delay
  decfsz delay0, f
  goto delay
  decfsz delay1, f
  goto delay
  return
shortdelay
  movlw 08h
  movwf delay1
  call delay
  clrf delay1
  return
randno
  clrf bitn
  clrf bitm
  btfsc randval, 5  ; Получаем n
  bsf bitn, 0       ; Если n = 1
  btfsc randval, 6  ; Получаем m
  bsf bitm, 0       ; Если m = 1
  movf bitn, w      ; n в w
```



**Листинг 8.3. Окончание**

```
  xorwf bitm, w     ; XOR m и n, результат - в w
  addlw H'00FF'     ; Установка разряда переноса, если w = 1
  rlf randval,f     ; Новое случайное число в randval
  return

end
```

**Листинг 8.4. Файл Toy02.hex**

```
:020000040000FA
:020000000528D1
:0800080005288B13831203137A
:1000100087018316870183120317 9E019F01031333
:1000200083122214503 0A50027208030220587003B
:10003000222 0A50B142887131C28A00B1D28A10B18
:100040001D2808000830A1001D20A1010800A301FF
:10005000A401A21A2314221B241423082406FF3E01
:04006000A20D0800E5
:02400E00C430BC
:00000001FF
```

Когда переменная `count` содержит значение 050h, длительность звукового пакета составляет около половины секунды. При большей длительности становятся слышны регулярные биения, которые портят эффект. Это обусловлено тем, что последовательность на самом деле не является случайной и повторяется через каждые 127 вызовов подпрограммы `randno`. Можно было бы расширить виртуальный сдвиговый регистр, скажем, до 16 разрядов и присваивать переменным `bitn` и `bitm` значения разрядов <15> и <14> соответственно. Последовательность при этом будет повторяться через каждые 32 767 вызовов.

# Глава 9. Робот “Искатель”

“Искатель” — это мобильная платформа, обеспечивающая возможность установки максимального числа датчиков и исполнительных механизмов, поддерживаемых микроконтроллером PIC 16F690. Этот робот идеален для экспериментов в области механики и электроники роботов, а также — робототехнического программирования.

На платформе достаточно места для нескольких датчиков, и ее можно легко расширить для размещения большего числа датчиков и исполнительных механизмов. Модульный дизайн электронных схем обеспечивает гибкость и возможность расширения. С микроконтроллером PIC в качестве центра управления робот “Искатель” готов к исследованию новых видов деятельности.

Спецификация робота:

- двухпалубная конструкция: нижний уровень — двигатели, колёса; верхний уровень — микроконтроллер PIC, датчики, исполнительные механизмы;
- две батареи: 12 В (двигатели), 6 В (логика);
- три колеса: два ведущих, один ролик;
- микроконтроллер PIC 16F690;
- датчики: переключатели, бамперы, фотоэлементы, инфракрасный;
- исполнительные механизмы: двигатели, светодиоды, зуммер/сирена;
- панель управления — на верхнем уровне.

Программы:

- программный селектор;
- “Скиталец” (обход препятствий);
- поиск света;
- отслеживание линии;
- “Узник” (нельзя пересекать линию);
- проход лабиринта.

## Механика

Палубы вырезаны из листа пластмассы (предпочтительно — ПВХ толщиной 3 мм), фанеры или твердого картона. Нижняя палуба робота



показана на рис. 9.1 и рис. 9.2. Вначале следует вырезать квадрат размерами 160×160 мм, а затем — углубления для приводных колес и срезать треугольные области на задних углах. Если используются колеса более 70 мм в диаметре, могут потребоваться большие углубления.

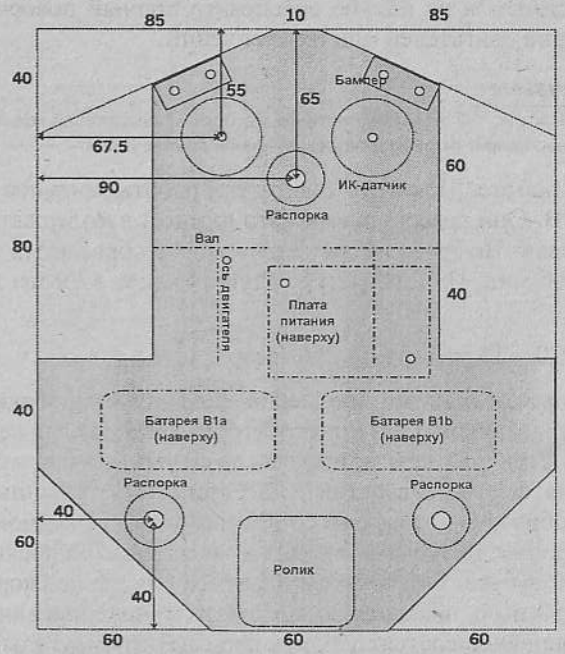



**Рис. 9.1.** Схема нижней палубы робота “Искатель” (вид снизу)

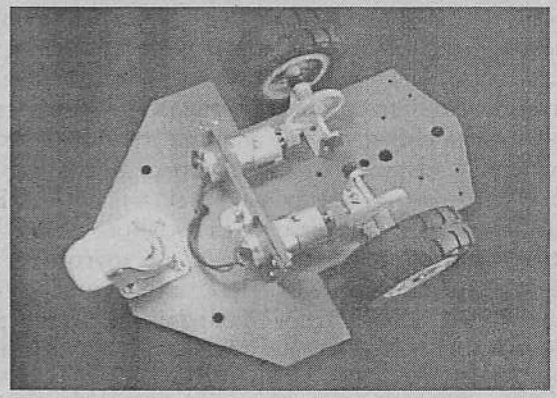


**Рис. 9.2.** Нижняя палуба робота “Искатель” (вид снизу)

Двигатели и передаточные механизмы установлены на нижней палубе (см. рис. 9.2). Таким образом, приводные колеса должны быть достаточно большими для обеспечения приемлемого дорожного просвета. На рис. 9.2 также показано основание ролика. При планировании размещения механизмов на палубе обеспечьте полный поворот ролика на 360° без касания двигателей или их креплений.

**Примечание**

На рис. 9.1 и рис. 9.2 показано положение осей и двигателей, однако точное их размещение зависит от размеров двигателей и других узлов.

В нашем роботе "Искатель" двигатели работают от постоянного напряжения 12 В. Они реверсируемы (это важно!) и содержат встроенные коробки передач. Выбранные двигатели могут сбрасывать частоту вращения до 70 об/мин. При диаметре ведущих колес в 70 мм это дает скорость робота:

$70 \times \Omega \times 70 = 15$ м/мин или 256 мм/с.

Такая скорость немного чрезмерна, поэтому мы добавили внешние передаточные механизмы (из конструктора Meccano) для ее дальнейшего снижения. При это нам пришлось применить игольчатый надфиль для изменения формы отверстия в шестерне ("скругленный треугольник") таким образом, чтобы оно соответствовало выходному валу двигателя. Шестерня с 10 зубьями входит в зацепление с плоским зубчатым колесом на 50 зубьев. Это дает пятикратное снижение скорости робота примерно до 50 мм/с, повышая возможности маневрирования.

Опоры для валов состоят из двух двойных прямоугольных скоб (по одной для каждого вала). Устанавливайте приводные узлы (двигатели, передаточные механизмы, опоры и колеса) так, чтобы палуба была горизонтальна земле, опираясь на приводные колеса и ролик.

Большинство двигателей имеют нарезные отверстия для болтов или же в них предусмотрены фиксаторы. Другие же не предоставляют никаких таких приспособлений и предназначены для установки в пружинном держателе. При невозможности обзавестись держателем требуемого размера можете изготовить монтажный узел из листа ПВХ. В таком случае каждый двигатель поддерживается с одного конца угловой скобой, через которую проходит выходной вал, а с другого — рамой, которая изготовлена из полос листового ПВХ, схваченных болтами, и зафиксирована на палубе робота угловой скобой (рис. 9.3 и рис. 9.4). Обмотайте тыльный конец каждого двигателя одним слоем изолирующей ленты, чтобы улучшить сцепление с ним рамы.



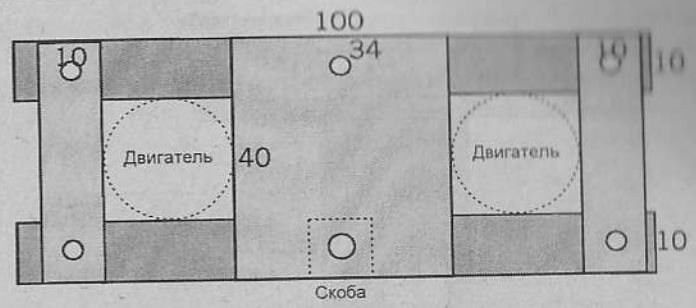



**Рис. 9.3.** Тыльные концы двигателей поддерживаются рамой

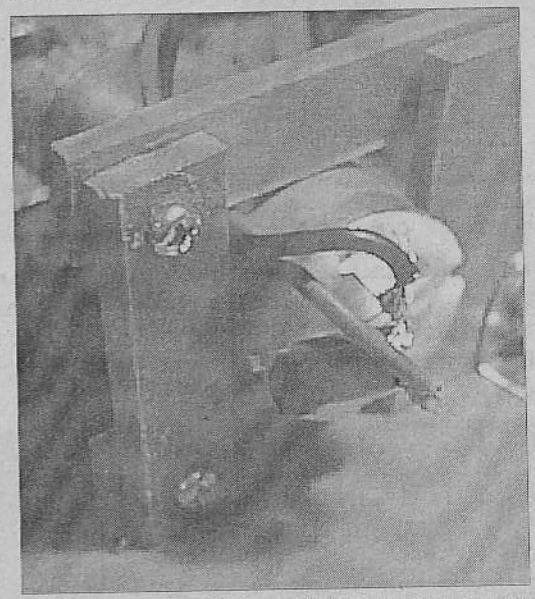


**Рис. 9.4.** Тыльные концы двигателей (со сглаживающим конденсатором, припаянным к клеммам) фиксируются рамой. Справа видна часть угловой скобы, удерживающей раму на палубе робота

Перед креплением чего-либо к палубе болтами просверлите в ней все отверстия. В основном, их диаметр — 3 мм или 4 мм, однако может оказаться, что три отверстия для распорок между палубами должны быть больше (рис. 9.5). Другие отверстия, не показанные на рис. 9.1, предназначены для установки скоб, фиксирующих оси, двигатели, раму и ролик. Два отверстия необходимы для установки платы питания, и еще два — для каждой петли бампера.

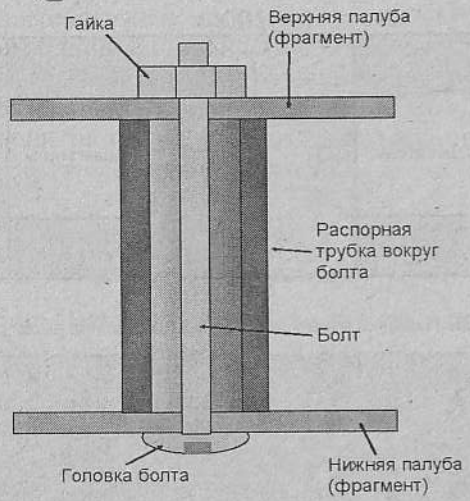



**Рис. 9.5.** Палубы соединяются тремя длинными болтами (здесь показан один). Каждый болт окружен распорной трубкой, вырезанной из шланга садового разбрызгивателя. Гайка зажимается таким образом, чтобы жестко удерживать палубы робота на концах трубки

Ролик (легкий мебельный) следует подобрать по размеру таким образом, чтоб после его фиксации болтами задний конец палубы располагался горизонтально земле (рис. 9.6). В противном случае придется корректировать или монтажный узел ролика, или узлы колес, чтобы выровнять палубу робота.

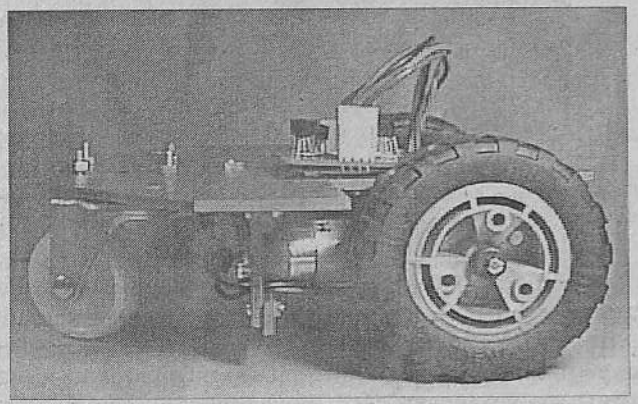


**Рис. 9.6.** Ролик — правильного размера, обеспечивая горизонтальное положение палубы робота. Здесь виден тыльный конец правого двигателя, поддерживаемого рамой, а также — плата питания, установленная наверху нижней палубы



Верхняя палуба робота (рис. 9.7) вырезана из квадрата ПВХ со стороной 160 мм. Ее общие габариты — те же, что и у нижней палубы. Она монтируется над нижней палубой таким образом, чтобы ее тыльный край находился в точности над тыльным краем нижней палубы.

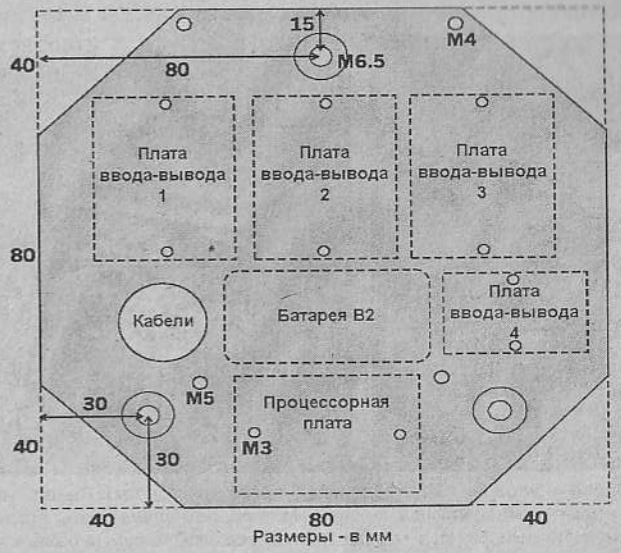



**Рис. 9.7.** Верхняя палуба с обозначением размещения плат и позиции трех распорок

Здесь необходимы три отверстия для распорных болтов, расположенные в точности в тех же позициях, что и отверстия в нижней палубе. Пространство между двумя палубами должно быть достаточным для обеспечения доступа рук и небольших инструментов, однако, если оно будет слишком большим, то возникнет вероятность опрокидывания робота на неровной поверхности. Вполне приемлемо расстояние в 100 мм.

Существуют и другие способы монтажа верхней палубы. Один из них — вырезать распорки из деревянной шпонки около 20 мм в диаметре или же использовать деревянный квадрат со стороной 20 мм. Палубы скрепляются путем пропускания винтов через отверстия в палубе и концах шпонки. Еще один вариант — скобы из алюминиевых полосок, прикрепленных болтами к палубам.

В верхней палубе есть большое отверстие (приблизительно 25 мм в диаметре) для кабелей, соединяющих две палубы. К их числу относится и четырехжильный кабель, идущий от платы процессора к плате управления двигателем. Отверстие должно быть достаточно большим, чтобы через него прошли разъемы, расположенные на обоих концах

этого кабеля. Альтернативно, можно было бы направить этот кабель другим путем. В таком случае отверстие может быть меньшим.

На рис. 9.8 показано размещение плат и батареи питания логических схем. Каждая плата фиксируется на палубе парой нейлоновых болтов.

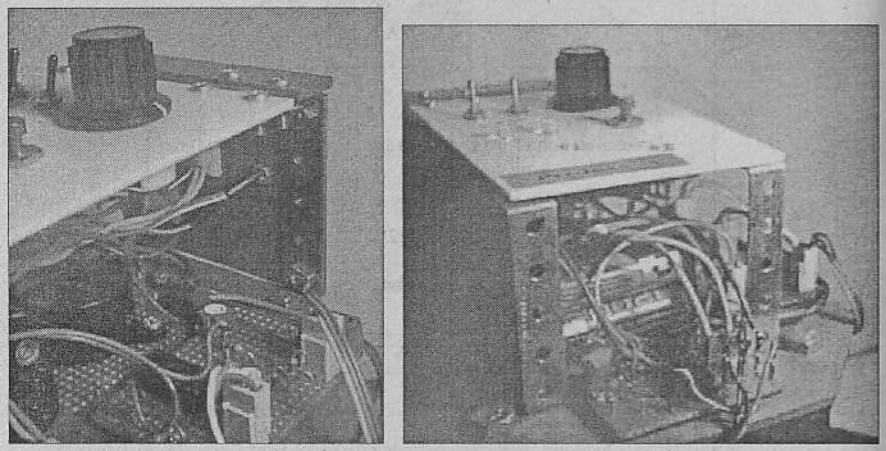


**Рис. 9.8.** Слева — верхняя палуба и панель управления (белая). Панель управления привинчена к раме из металлической рейки. На переднем плане видны платы бамперов и инфракрасные датчики. Справа — вид сзади на верхнюю палубу и панель управления. Тыльный край панели поддерживается двумя вертикальными металлическими рейками (но не крепится к ним), привинченными болтами к палубе. На переднем плане видна плата процессора, а за ней — батарея на 6 В

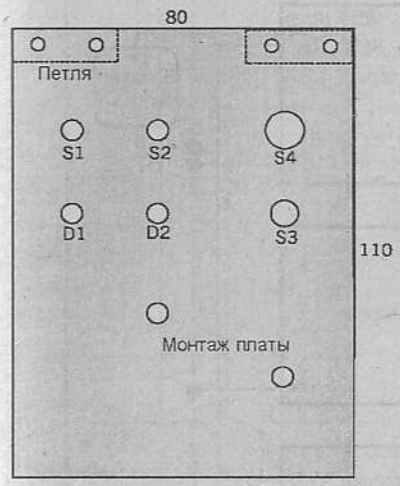
Для каждой платы, которая должна быть включена в систему, просверлите пару отверстий в палубе, соответствующих отверстиям в платах. Обратите внимание, что в палубах нет никаких отверстий для фиксации батарей, поскольку они крепятся самоклеющейся лентой. Это облегчает их снятие для замены или зарядки элементов.

На рис. 9.8 видно, что одна из двух вертикальных стоек крепится болтами к верхней палубе робота. Панель управления крепится на петлях к поперечной ПВХ-планке на верхних концах опор. Сзади панель управления опирается на другую пару опор.

Пространство между верхней палубой робота и панелью управления заполнено схемами и монтажными проводами, однако панель управления может подниматься, обеспечивая свободный доступ к компонентам. Соединения будет легче отслеживать, если использовать разноцветные провода. Панель управления — это прямоугольник из белого листового полистирола, однако ее можно было бы изготовить и из ПВХ.



На панели установлены переключатели S1 и S2 для батареи B1 (приводные двигатели на 12 В) и батареи B2 (логические схемы, 6 В), а также их индикаторные светодиоды: D1 и D2 (рис. 9.9). Кнопка S3 сбрасывает систему. Поворотный переключатель S4 выбирает, какая из четырех программ должна быть запущена. На панели управления могли бы присутствовать и другие переключатели, активизирующие определенные схемы поведения робота.



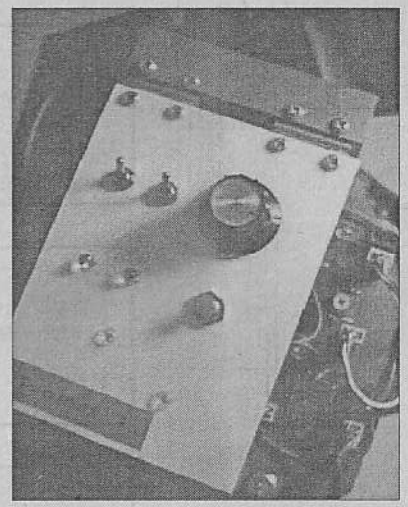



**Рис. 9.9.** Панель управления с двумя отверстиями для монтажа платы управления

Вот, в основном, и все, что можно сказать о конструкции данного мобильного робота. Можно реализовать множество его версий, заменяя некоторые материалы и используя другие методы сборки. В конечном счете полученная конструкция может выглядеть совершенно по-другому, однако ко всем мобильным роботам применимы одни и те же принципы. Экспериментируйте и изобретайте!

## Электроника

После общего обзора рассмотрим электронную часть робота "Искатель" — плата за платой (рис. 9.10). Список необходимых компонентов дан в конце этого раздела.

Центром системы является плата процессора, на которой установлен микроконтроллер PIC 16F690. Питание на него поступает от B2 через переключатель S2, размещенный на панели управления. Включение питания индицирует светодиод D2.

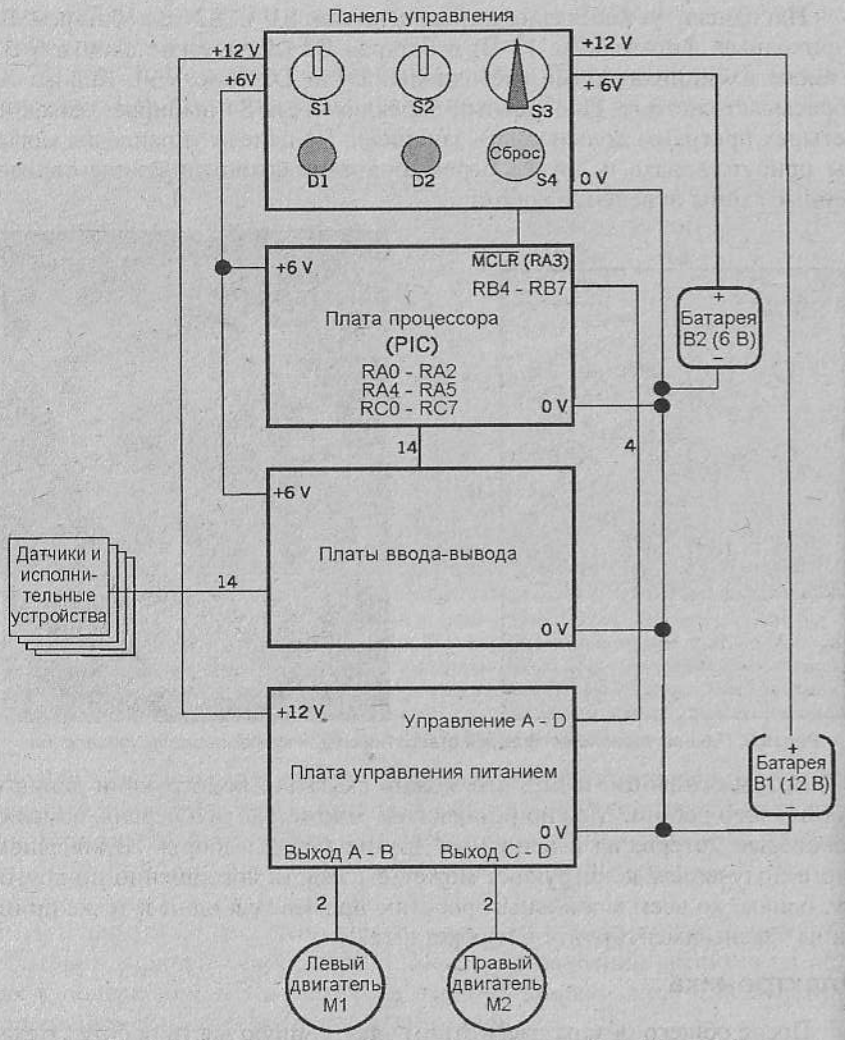



**Рис. 9.10.** Структурная схема электронной части робота "Искатель"

На плате управления размещены подключения к S3 и S4, к плате управления питанием, расположенной на нижней палубе робота, а также — к платам ввода-вывода на верхней палубе и всем другим модулям, реализующим взаимодействие датчиков и исполнительных устройств с микроконтроллером PIC.



### Плата контроллера

По рис. 9.11 видно, что эта плата содержит лишь микроконтроллер PIC и схему для подключения подтягивающего сопротивления ко входному каналу RA3 (вывод 4), не имеющего внутреннего подтягивающего резистора.

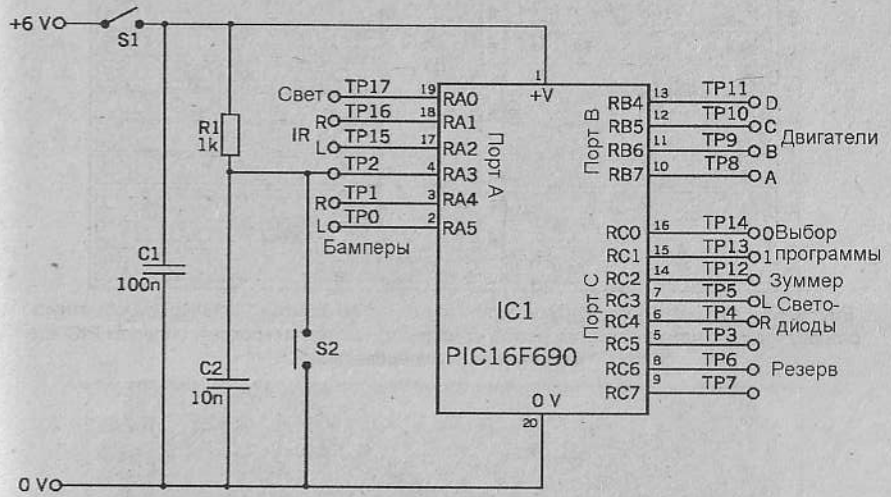



**Рис. 9.11.** Схема платы контроллера

Выключатель питания S2 и кнопка S3 размещены на панели управления. Как видно из рис. 9.12 и рис. 9.13, основной элемент этой платы — 20-контактное гнездо для установки микроконтроллера PIC. На ней также присутствует большое число печатных выводов, предназначенных для ввода-вывода.

Задействованная здесь методика межплатных соединений обеспечивает максимальную гибкость при конфигурировании системы на различные наборы датчиков и исполнительных устройств.

На рис. 9.12 показаны два монтажных отверстия M3, просверленных в позициях G2 и G19. Плата привинчивается болтами непосредственно к верхней палубе робота с помощью нейлоновых болтов и гаек M3. По окончании монтажа платы единственное, что требуется проверить, — непрерывность соединений между отдельными гнездами разъема микроконтроллера и соответствующими выводами.

Отдельные выводы для каждого соединения обеспечивает гибкость, однако в результате необходимо использовать двух- или четырехконтактные разъемы для подключения некоторых плат ввода-вывода.

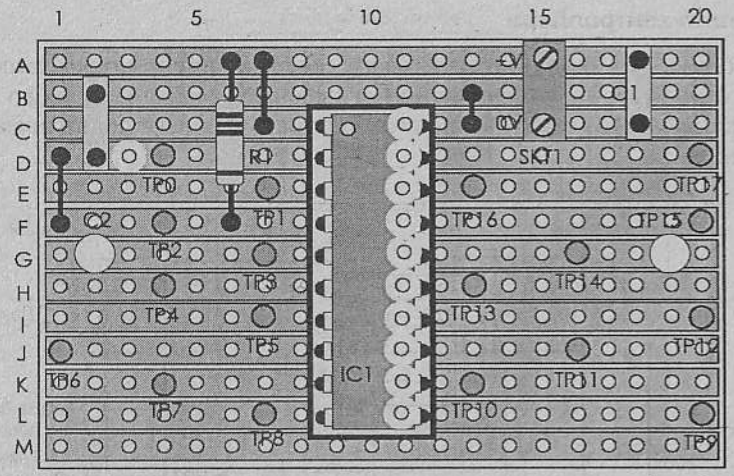



**Рис. 9.12.** Разводка платы контроллера. По окончании монтажа исследуйте обратную сторону платы с помощью лупы, чтобы убедиться, что под микроконтроллером PIC все медные полоски перерезаны

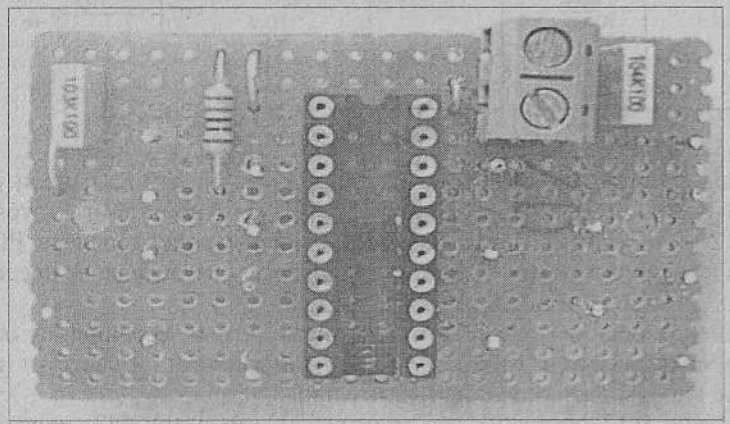


**Рис. 9.13.** Завершенная плата контроллера. Микроконтроллер PIC не установлен в гнездо, чтобы продемонстрировать позолоченные контакты. Этот тип разъемов лучше всего подходит для установки микроконтроллера, который необходимо многократно извлекать для перепрограммирования во время разработки и тестирования системы

### Плата управления электромотором

Эта схема представляет собой обычный Н-образный мост. Разводка платы управления двигателем показана на рис. 9.14, а ее завершенный вид — на рис. 9.15.



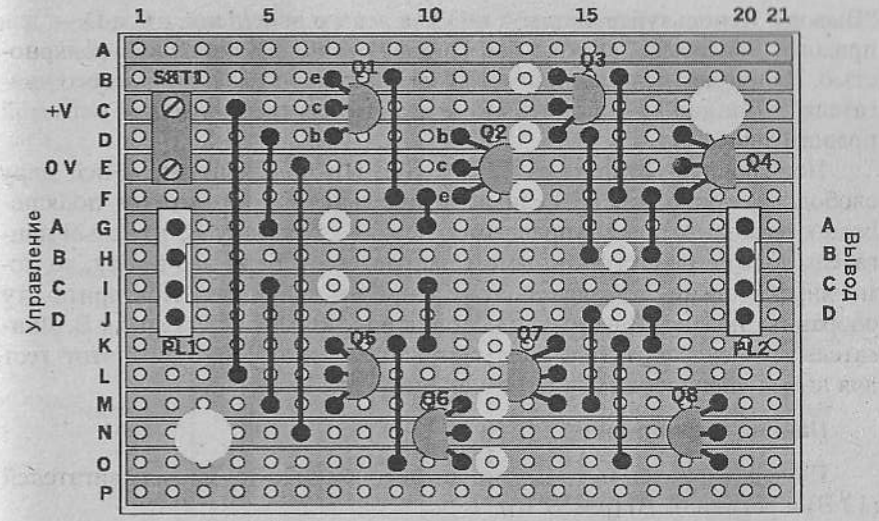



**Рис. 9.14.** Разводка платы управления двигателем

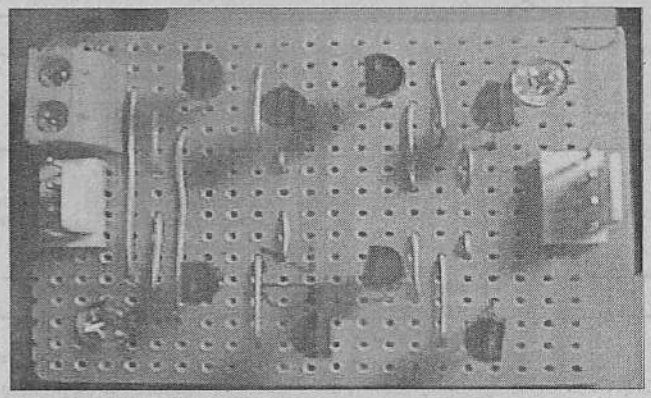


**Рис. 9.15.** Завершенная плата управления двигателем

Эта версия схемы обеспечивает независимую работе двух двигателей. Подключение к микроконтроллеру PIC (ввод сигналов управления) обеспечено через четырехконтактный разъем, показанный на рис. 9.15 слева. Вывод для двух двигателей осуществляется через другой четырехконтактный разъем, показанный на рис. 9.15 справа.

Проверьте завершенную плату обычным способом, исследовав непрерывность соединений и отсутствие короткозамкнутых цепей. Для функционального теста временно подключите двигатели к контакту

"Вывод". Используйте линии A и B для левого двигателя, а C и D — для правого. Двигатели должны быть подключены с одинаковой полярностью. Например, если линия A связана с некоторой клеммой левого двигателя, то линия C должна быть связана с соответствующей клеммой правого двигателя.

Подключите батарею на 12 В к SKT1 и подсоедините к ней пару свободных проводников. Если положительная линия питания подключена к выводу A платы управления, а 0 В — к выводу B, то левый двигатель должен работать в прямом направлении. Если это не так, то поменяйте местами подключения на клеммах двигателя. Повторите эту операцию, подав 0 В на A и подключив положительный полюс к B. Двигатель должен работать в обратном направлении. Повторите этот тест для линий управления C и D.

### Панель управления

Панель управления коммутирует источники питания для двигателей (12 В) и логики (6 В) (рис. 9.16).

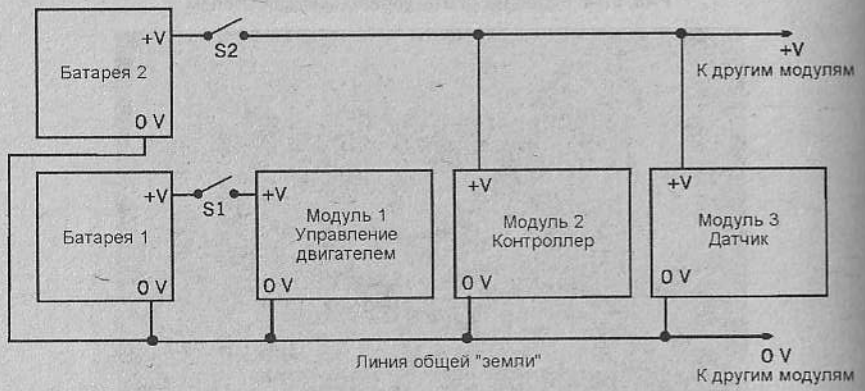



**Рис. 9.16.** Схема распределения питания. Переключатели S1 и S2 находятся на панели управления

Обратите внимание на линию 0 В, соединяющую клеммы 0 V обеих батарей с выводами 0 V платы процессора и всеми платами датчиков и исполнительных устройств. Таким образом, у всех модулей в системе один уровень "земли" (0 В). Модули могут обмениваться сигналами.

Панель управления содержит несколько независимых цепей (рис. 9.17):

- переключатели питания S1 и S2 с соответствующими индикаторными светодиодами D1 и D2;



- поворотный переключатель выбора программы S3, выход которого соединен с платой контроллера;
- кнопка S4, выход которой соединен с платой контроллера.

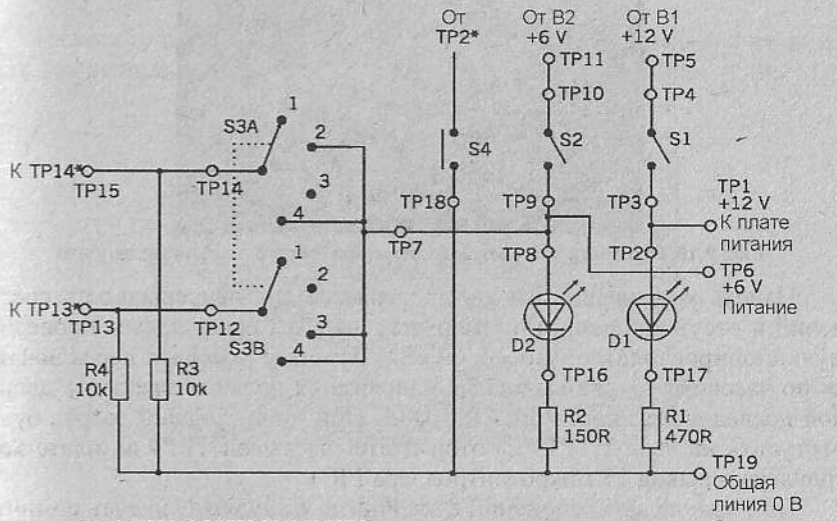



**Рис. 9.17.** Схема панели управления

Разводка платы для панели управления показана на рис. 9.18, а завершенная плата — на рис. 9.19.

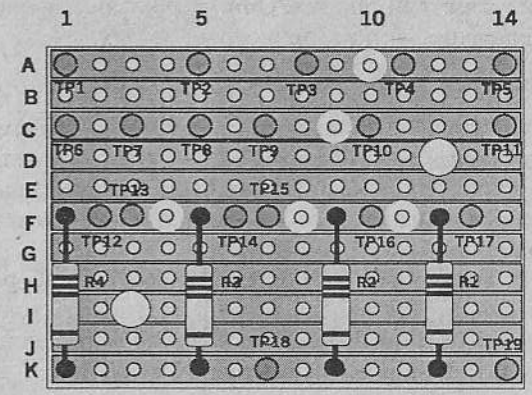



**Рис. 9.18.** Основная часть соединений панели управления — между компонентами, установленными на панели. Также в ней размещена небольшая плата с полосками, соединяющая компоненты панели с внешними модулями с помощью выводов

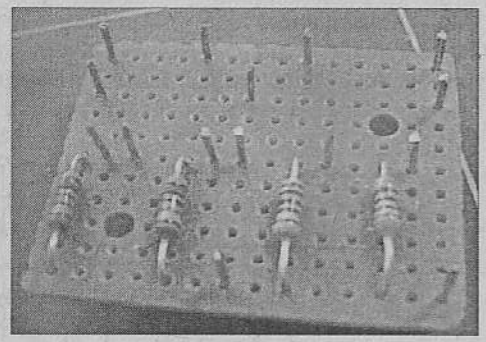


**Рис. 9.19.** Эта плата установлена с обратной стороны панели управления

Панель управления быстро проверяется для непрерывность соединений и отсутствие короткозамкнутых цепей. Также следует проверить функционирование схем (особенно S4). По мере поворота переключателя по часовой стрелке выходные напряжения должны следовать двоичной последовательности 00, 01, 10, 11. При этом старший разряд будет поступать на вывод TP13 на этой плате (на вывод TP13 на плате контроллера и вывод 15 микроконтроллера PIC).

При прокладке соединений с внешними модулями следует помнить, что провода должны проходить вблизи края панели, посаженного на петли. В противном случае будет трудно поднять панель при тестировании или обслуживании системы. Мы использовали небольшие самоклеящиеся пластмассовые кабельные держатели, зафиксированные на тыльной стороне передней панели. Они удерживают провода вблизи от края панели управления, установленного на петлях.

### Модули ввода-вывода

У робота "Искатель" пять входных модулей: фотоэлемент, пара инфракрасных датчиков и пара бамперов. Он также получает сигнал от S3 и S4 на панели управления, как это уже было отмечено выше. Если исключить приводные двигатели, то у робота три выходных модуля: зуммер и два светодиода высокой яркости.

Оборудовать свой робот в точности таким же набором устройств совсем необязательно. Для начала можете установить только бамперы. Поэкспериментируйте с ними некоторое время перед добавлением некоторых других датчиков и исполнительных механизмов. Другие датчики и исполнительные схемы, которые могли бы расширить диапазон поведения робота, рассматриваются в главе 3. Некоторые поставщики робототехнических комплектов также продают готовые модули датчиков



и исполнительных устройств, которые можно подключить к роботу "Искатель". Испробуйте некоторые из них.

### Фотоэлемент и светодиоды

Поскольку на верхней палубе робота есть место только для четырех плат, мы решили собрать эти два модуля на одной плате (рис. 9.20).

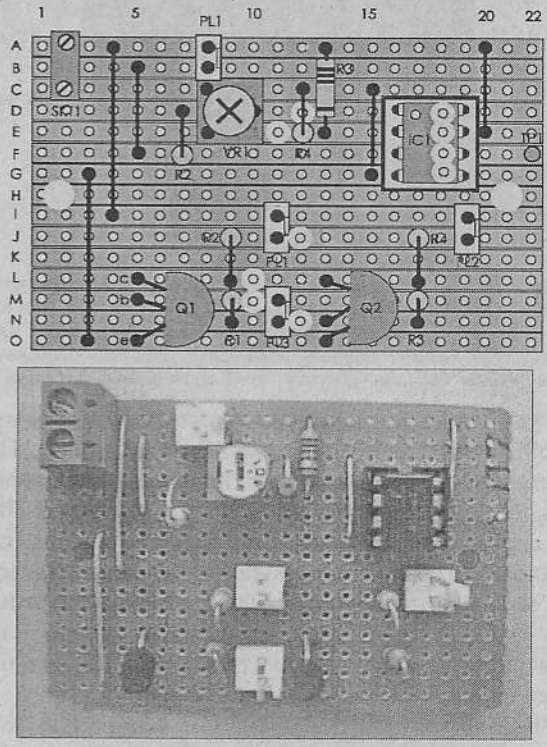



**Рис. 9.20.** Фотоэлемент занимает верхнюю часть платы, а пара переключателей светодиодов находится в нижней ее части

Обычно в качестве фотоэлемента используется фоторезистор типа ORP12, однако его можно заменить любым недорогим фоторезистором. Предпочтительно, чтобы он был около 5 мм в диаметре. Фоторезистор имеет два вывода, припаянных к проводникам, длиной около 60 мм, которые заканчиваются двухконтактным вставным разъемом. Полярность значения не имеет. Плата занимает позицию 1 (см. рис. 9.7) в правом переднем углу верхней палубы. Фоторезистор установлен на лицевой панели (рис. 9.21).

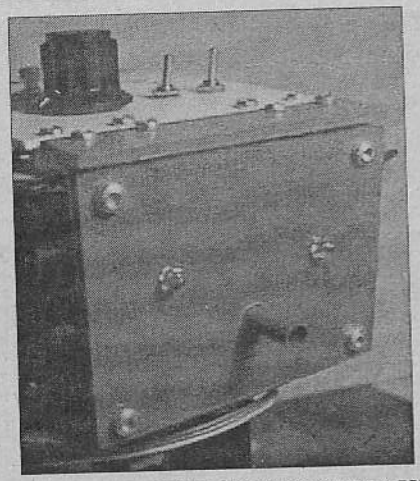


**Рис. 9.21.** Лицевая панель с установленным фотоэлементом и двумя светодиодами. Она вырезана из ПВХ по габаритам передних опор панели управления. Для обеспечения направленной чувствительности на фоторезистор достаточно плотно надет отрезок пластиковой непрозрачной трубки длиной 10 мм, который затем пропускается в отверстие в панели (посадка при этом также должна быть плотной)

Светодиоды устанавливаются путем сверления двух отверстий диаметром 1 мм с разносом в 2 мм для каждого светодиода. Провода выводов пропускаются через эти отверстия, на основание каждого светодиода наносится клей, и светодиод приклеивается к панели. Выводные провода каждого светодиода припаиваются к паре проводов приблизительно 100 мм длиной, которая завершается двухконтактным вставным разъемом. При этом важна полярность. Катод LED1 должен идти на вывод I11, а анод — на J11. Катод D2 идет на I19, а анод — на J19. Катод обычно идентифицируют тем, что его вывод расположен ближе к ободу на корпусе светодиода.

Обратите внимание на ПВХ-рейку, прикрепленную болтами поверх петель. Ее назначение — прикрыть зазор между передним краем панели управления и верхним краем лицевой панели.

Датчик проверяется путем подачи напряжения 6 В на SKT1 и подключения вольтметра к TP1 и 0 В. Корректируйте настройку VR1 до тех пор, пока выходной сигнал не примет состояние низкого уровня (менее 1 В), когда фоторезистор прикрыт, и состоянию высокого уровня (более 3,5 В), когда фоторезистор (в трубке) подвергается воздействию дневного света или настольной лампы. Весьма вероятно, что при испытаниях робота эту настройку придется откорректировать в зависимости от условий окружающего освещения.



При подключенном питании платы и светодиодами, подключенными к PL1 и PL2, светодиоды проверяются путем подачи 6 В на вывод N11. Светодиод D1 должен при этом включится. Точно так же проверьте включение D2, подав 6 В на вывод M11.

### Плата инфракрасных датчиков

Инфракрасные датчики в роботе "Искатель" установлены под нижней палубой (сразу позади каждого бампера) и направлены вниз. Платы ИК-датчиков — небольшие, с полосками (рис. 9.22 и рис. 9.23). Экранирующие обода находятся на уровне около 20 мм над землей (рис. 9.24).

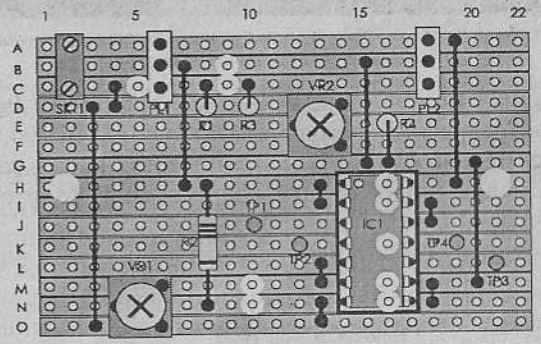



**Рис. 9.22.** Плата инфракрасных датчиков связывает с системой два датчика. Она оснащена двумя разъемами: PL1 — для подключения левого датчика, PL2 — правого. Подключения: полоса A — + 6 В; полоса B — выход катода светодиода; полоса C — от анода инфракрасного светодиода. Выходной сигнал обычно снимается с гнезд TP1 (левый) и TP3 (правый)

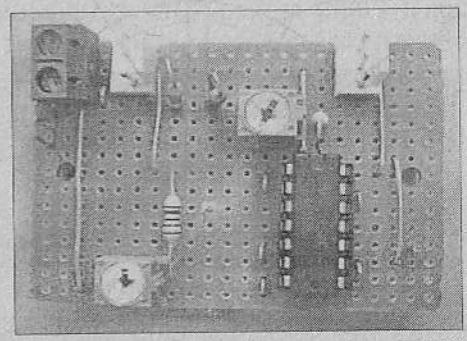


**Рис. 9.23.** Плата инфракрасных датчиков (схема — см. рис. 3.23). Как и для платы с фотоэлементами, она оснащена микросхемой компаратора. Вместо него можно было бы использовать и встроенный модуль компаратора микроконтроллера PIC или АЦП, однако уровни переключения при этом должны были бы устанавливаться программно


**Рис. 9.24.** Вид робота снизу, показывающий два инфракрасных датчика. Видны также платы и белые пластмассовые экраны

Инфракрасный светодиод (D1) — диаметром 5 мм (максимальный ток 50 мА). Инфракрасный фотодиод (D2) — типа BP104, который размещается максимально близко к плате, но могут использоваться и другие типы. При пайке диодов обратите внимание на то, что они устанавливаются с противоположной полярностью.

Платы датчиков охвачены ориентированными вниз экранами (рис. 9.25).

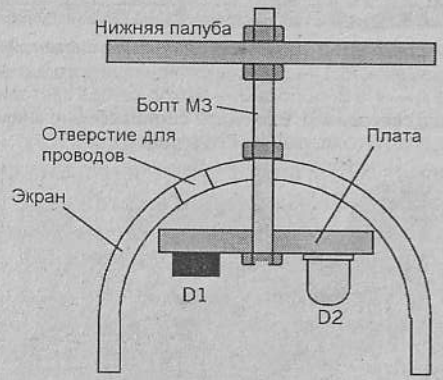


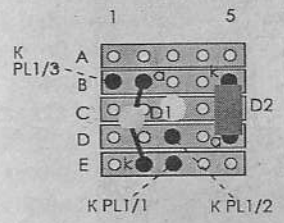

**Рис. 9.25.** Поперечный разрез зонда. Внутри экран может быть окрашен в матово-черный цвет, чтобы дать большую направленную чувствительность, однако было обнаружено, что на самом деле разница невелика

Мы использовали ПВХ-колпачки для труб, купленные в отделе сантехники местного хозяйственного магазина. Болт М3 длиной 30 мм проходит сквозь плату через центральное отверстие, просверленное в экране и через нижнюю палубу робота. Эти компоненты зафиксированы



гайками. Второе отверстие сверлится в экране сбоку. Оно предназначено для монтажных проводов (на рис. 9.25 не показаны), которые проходят через плату с ее тыльной стороны, а затем припаиваются.

Смонтированные датчики показаны на рис. 9.26. Из центры разнесены примерно на 45 мм.



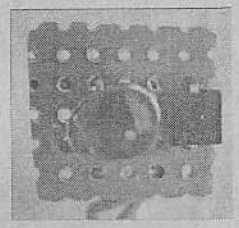



**Рис. 9.26.** Слева — разводка платы с полосками для инфракрасных датчиков. В ней есть центральное отверстие для крепежного болта. Справа — выводы D1 коротко обрезаны, чтобы проходили близко к плате

Датчики тестируются с помощью куска белого картона, половина которого окрашена в черный цвет. Подайте напряжение 6 В и включите один из датчиков. Подсоедините вольтметр к выходу. Откорректируйте установочный резистор так, чтобы выходное напряжение соответствовало состоянию логического нуля (около 0 В), когда белая область картона удерживается на расстоянии 20 мм от датчика, но повышалось до положительного значения напряжения питания при размещении перед датчиком черной половины картона.

### Бамперы

Это две прямоугольные ПВХ-панели (левая и правая), смонтированные на петлях перед нижней палубой робота (рис. 9.27).


**Рис. 9.27.** Микропереключатель привинчен болтами к маленькой ПВХ-панели, подвешенной под нижней палубой на двух болтах. Здесь правый бампер откинут назад

Они висят вертикально, однако при нажатии на них отклоняются назад и вызывают срабатывание микропереключателей (рис. 9.28). Бамперы должны быстро возвращаться в исходное положение, когда давление на них исчезает, поэтому петли должны двигаться свободно.

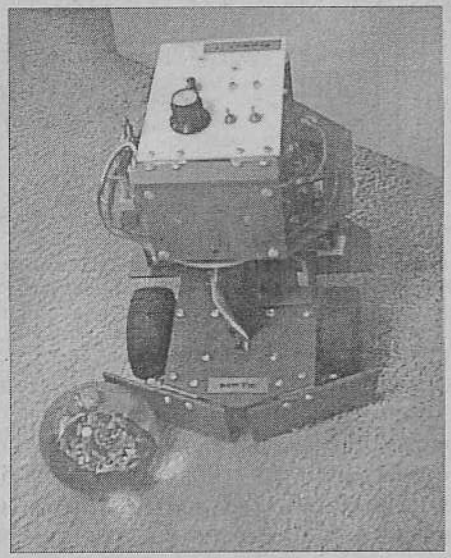


**Рис. 9.28.** Робот встретил препятствие. Он распознал, что его правый бампер на что-то наткнулся, и сработал правый микропереключатель. Обычно бамперы размещают спереди робота, однако могут пригодиться и задние бамперы для обнаружения препятствий при движении назад. Бампер или его аналог может также устанавливаться сбоку робота для использования в подпрограммах отслеживания стен

Разводка платы бампера показана на рис. 9.29, а ее внешний вид — на рис. 9.30.

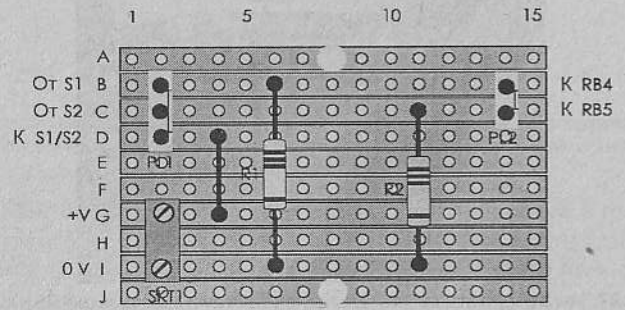



**Рис. 9.29.** Плата бамперов. Номинал резисторов — 10 кОм



### Плата зуммера

Плата зуммера (рис. 9.31) или другого устройства звуковой сигнализации может быть установлена в наименее доступной позиции на верхней палубе (рис. 9.32), поскольку на этой плате нет ничего, что будет нуждаться в корректировке. Разводка платы с полосками позволяет адаптировать размещение компонентов для устройств различных форм и размеров.

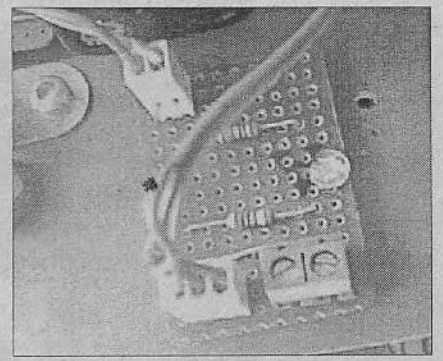


**Рис. 9.30.** Плата бамперов, установленная в позиции 4 на верхней палубе робота

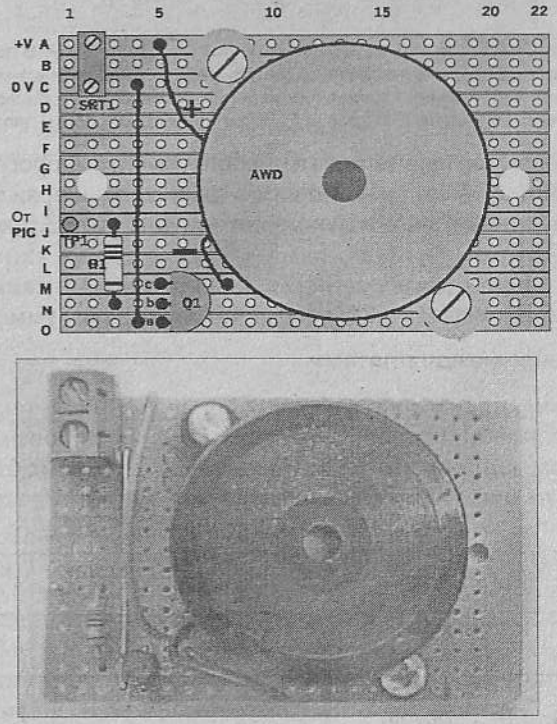



**Рис. 9.31.** Плата зуммера. Для получения максимальной громкости удостоверитесь, что устройство звуковой сигнализации жестко зафиксировано на плате, а плата — на палубе

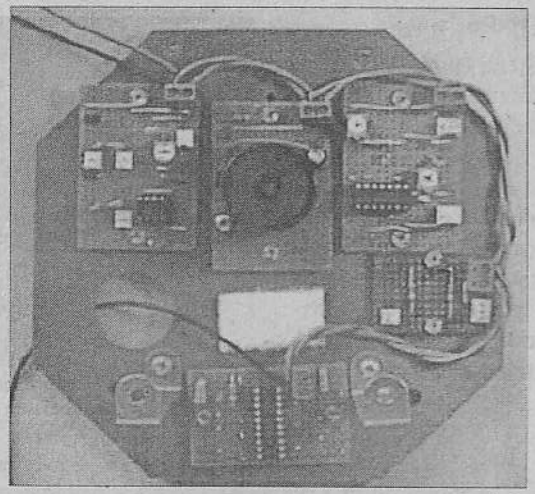


**Рис. 9.32.** Теперь мы готовы к реализации межплатных соединений. В первую очередь подайте напряжение питания на верхнюю палубу робота. Обратите внимание на линии питания (слева вверху), идущие к панели управления, а также — на общую линию "земли", идущую к нижней палубе. Батарея на 6 В фиксируется полоской липкой ленты.

Для звуковой сигнализации мы использовали недорогой пьезоэлектрический мини-зуммер, работающий в диапазоне напряжений от 1,5 В до 28 В. Он воспроизводит только один тон частотой 3,5 кГц и громкостью около 80 дБ.

Проверьте плату, подключив к ней напряжение питания 6 В. Затем приложите + 6 В к выводу TP1, чтобы услышать звук зуммера.

### Соединения между платами

Соединения иллюстрируют рис. 9.10, рис. 9.16, а также табл. 9.1. Рис. 9.32 показывает способ последовательного подключения линий 0 В и +6 В от платы к плате. Микроконтроллер PIC в разъеме отсутствует до тех пор, пока не будет проверена вся система.

**Таблица 9.1.** Соединения между платами робота "Искатель"

| От платы | К | Проводов | От соединителя | К соединителю |
|---|---|---|---|---|
| Панель управления | Процессор MCLR | 1 | Вывод* | Вывод |
| Панель управления | Процессор Выбор программы | 2 | Припаян к S4 | Выводы |



**Таблица 9.1.** Окончание

| От платы | К | Проводов | От соединителя | К соединителю |
|---|---|---|---|---|
| Фотоэлемент | Фоторезистор | 2 | Двухконтактное гнездо** | Припаян к фоторезистору |
| Фотоэлемент | Процессор | 1 | Вывод | Вывод |
| Светодиоды | Процессор | 2 + 2 | Двухконтактные гнезда | Припаяны к светодиодам |
| Инфракрасный датчик | Инфракрасные датчики | 3 + 3 | Трехконтактные гнезда** | Припаяны |
| Инфракрасный датчик | Процессор | 2 | Выводы | Выводы |
| Бамперы | Бамперы | 3 | Трехконтактное гнездо | Припаяны |
| Бамперы | Процессор | 2 | Двухконтактное гнездо | Выводы |
| Зуммер | Процессор | 1 | Вывод | Вывод |

Примечания:

* Вывод — 0,9 мм на плате, гнездо на проводе.

** Двухконтактное и трехконтактное — полярные гнезда на плате, разъем на проводнике.

Начинайте тестирование с проверки питания каждой платы. Положительная линия на платах должна быть под напряжением 6 В или от 4,8 В до 5 В для перезаряжаемых элементов. Проверьте процессы ввода-вывода (например, что происходит при подаче заданного напряжения к выходу в разъеме процессора или какое напряжение на выходе при стимулировании датчика?).

### Выводы микроконтроллера PIC

В табл. 9.2 перечислены подключения к выводам микроконтроллера PIC. Ксерокопия этой таблицы, прикрепленная к стенду, будет полезной при сборке и тестировании модулей системы.

Строки табл. 9.2 сгруппированы по портам ввода-вывода. В ней перечислены все каналы, присутствующие в PIC16F690. Для порта B в крайнем правом столбце перечислены двоичные значения, используемые в ассемблерном коде для получения требуемого движения робота. Если оно не получено, то поменяйте местами подключения между платой управления питанием и клеммами двигателя.

Четырехпозиционный переключатель выбора программы с двумя входными линиями необязателен, если микроконтроллер выполняет

только одну или две программы. Вместо этого линии выбора програм можно использовать для других целей (например, подключения мик переключателей для дополнительных бамперов или для концевых ключателей).

**Таблица 9.2.** Подключения к выводами микроконтроллера

| Порт | Вывод | Номер | Тип | Подключение | 0 = | 1 = |
|---|---|---|---|---|---|---|
| A | RA0 | 19 | Вход | Фотоэлемент | Темно | Свет |
| | RA1 | 18 | Вход | ИК-датчик (правый) | Белый | Черн |
| | RA2 | 17 | Вход | ИК-датчик (левый) | Белый | Черн |
| | RA3 | 4 | Вход | Кнопка | Нажата | Отпу щена |
| | RA4 | 3 | Вход | Бампер (правый) | Нет кон-такта | Конта |
| | RA5 | 2 | Вход | Бампер (левый) | Нет кон-такта | Конта |
| B | RB4 | 13 | Выход | Двигатели, D (лев.) | A0 — вперед | |
| | RB5 | 12 | Выход | Двигатели, C (лев.) | 50 — назад | |
| | RB6 | 11 | Выход | Двигатели, B (прав.) | 60— вправо | |
| | RB7 | 10 | Выход | Двигатели, A (прав.) | 90 — влево | |
| C | RC0 | 16 | Вход | Выбор программы 0 | | |
| | RC1 | 15 | Вход | Выбор программы 1 | | |
| | RC2 | 14 | Выход | Зуммер | Выкл. | Звучит |
| | RC3 | 7 | Выход | Светодиод (правый) | Выкл. | Вкл. |
| | RC4 | 6 | Выход | Светодиод (левый) | Выкл. | Вкл. |
| | RC5–7 | 5, 8, 9 | Резерв | | | |

## *Список приобретений для электронной части робота*

Плата контроллера:

- R1 — резистор на 1 кОм;
- C1 — конденсатор с полиэфирным диэлектриком на 100 нФ;
- C2 — конденсатор с полиэфирным диэлектриком на 10 нФ;
- 20-контактный разъем с двухрядным расположением выводов;
- двухконтактная винтовая клемма;
- печатные выводы 0,9 мм (17 шт.);
- плата с полосками: 13 полос x 20 отверстий.

Плата управления двигателем (сдвоенная):

- Q1, Q3, Q5, Q7 — n-p-n-транзистор BC639 (4 шт.);
- Q2, Q4, Q6, Q8 — p-n-p транзистор BC640 (4 шт.);
- четырехконтактные разъемы (2 шт.);



- двухконтактная винтовая клемма;
- плата с полосками: 16 полос x 21 отверстие.

Панель управления:

- R1 — резистор на 470 Ом;
- R2 — резистор на 150 Ом;
- R3, R4 — резисторы на 10 кОм;
- D1, D2 — светодиоды, 5 мм;
- S1, S2 — миниатюрные двухпозиционные переключатели;
- S3 — миниатюрный трехполюсный поворотный электронный переключатель на четыре позиции;
- S4 — кнопка;
- печатные выводы 0,9 мм (19 шт.);
- плата с полосками: 11 полос x 14 отверстий.

Плата датчика освещенности:

- R1 — фоторезистор ORP12 или подобный;
- R2 — резистор на 470 Ом;
- R3, R4 — резисторы на 10 кОм (2 шт.);
- VR1 — миниатюрный подстроечный потенциометр (триммер) на 10 кОм;
- IC1 — операционный КМОП-усилитель CA3140E;
- восьмиконтактный разъем для микросхемы с двухрядным размещением выводов;
- винтовая двухконтактная клемма;
- двухконтактный разъем и штекер;
- печатный выводы 0,9 мм;
- плата с полосками: 15 полос x 22 отверстия.

Переключатели светодиодов (на плате фотоэлементов):

- R1, R3 — резисторы на 470 Ом;
- R2, R4 — резисторы на 120 Ом;
- Q1, Q2 — n-p-n-транзисторы BC548 или подобные;
- D1, D2 — светодиоды 5 мм, высокой яркости (2 шт.).

Плата инфракрасных датчиков:

- R1, R3 — резисторы на 68 Ом (2 шт.);
- R2, R4 — резисторы на 22 кОм (2 шт.);
- VR1, VR2 — миниатюрные подстроечные потенциометры (триммеры) на 100 кОм (2 шт.);
- D1, D3 — инфракрасные светодиоды, 5 мм (2 шт.);
- D2, D4 — инфракрасные фотодиоды BP104 или подобные (2 шт.);
- IC1 — КМОП-микросхема серии 4011, четыре вентиля "НЕ-И";

двухконтактная винтовая клемма;
трехконтактные разъемы и штекеры (2 шт.);
14-контактное гнездо для микросхемы с двухрядным размещением выводов;
печатные выводы 0,9 мм (4 шт.);
плата с полосками: 15 полос x 22 отверстия;
материал для экранов.

Плата бамперов:

R1, R2 — резисторы на 10 кОм (2 шт.);
двухконтактная винтовая клемма;
двухконтактный разъем и штекер;
трехконтактный разъем и штекер;
MS1, MS2 — миниатюрные микропереключатели на одно или два направления (2 шт.);
плата с полосками: 10 полос x 15 отверстий.

Плата зуммера:

R1 — резистор на 470 Ом;
Q1 — n-p-n-транзистор BC548 или подобный;
полупроводниковый зуммер AWD или сирена, работающие от 6 В;
двухконтактная винтовая клемма;
вывод 0,9 мм;
плата с полосками: 15 полос x 22 отверстия;
болты для фиксации зуммера (2 шт.);

Разное:

монтажный провод;
припой;
болты для фиксации печатных плат.

## Программирование

В этом разделе подразумевается, что используется программатор PICkit 2 и его программное обеспечение. В случае использования другого программатора листинги программ, в основном, будут такими же лишь с незначительными отличиями.

Первые командные строки программы определяют переменные, однако перед этим желательно создать "шапку", а также необходимо указать используемый процессор и определить его конфигурацию. Листинг 9.1. написан для микроконтроллера 16F690. Конфигурационное слово имеет значение `0x33c4` (см. главу 4). Обратите внимание на то, что директива `__config` начинается двумя символами подчеркивания. Листинг продолжается директивами, определяющими адреса регистров, метки `w` (рабочий регистр) и `f` (текущий файловый регистр), а также метки подпрограмм задержки.



Определены также дополнительные метки для переменных, используемых в режимах 2 и 4. Если данные режимы не используются эти метки можно не определять.

**Листинг 9.1. Начало программы для робота "Искатель"**

```
;*************************************************************

; Работа в одном из четырех выбираемых режимах:
; 1) "Скиталец" - используются бамперы для обхода препятствий
; 2) Поиск света
; 3) Отслеживание линии
; 4) "Узник" - робот движется случайным образом внутри области,
;    обведенной линией

;*************************************************************

  list        p=16F690
  __config    0x33c4

  status      equ 03h
  porta       equ 05h
  portb       equ 06h
  portc       equ 07h
  trisa       equ 05h
  trisb       equ 06h
  trisc       equ 07h
  ansel       equ 1eh
  anselh      equ 1fh

  w           equ 00h
  f           equ 01h
  z           equ 02h

  delay0      equ 020h
  delay1      equ 021h
  delayn      equ 022h

; Директива для режима 2
  scans       equ 023h

; Директива для режима 4
  randval     equ 024h
  bitn        equ 025h
  bitm        equ 026h
```

**Листинг 9.1. Окончание**

```
; Программа начинается здесь

  goto start
  org H'0004'
  goto start

start
  bsf status, 5      ; Страница 1
  clrf trisb         ; Все разряды порта B - выходы
  movlw H'03'        ; RC0 и RC1 - входы
  movwf trisc
  bcf status, 5      ; Назад на страницу 0
  bsf status, 6      ; Страница 2
  clrf ansel         ; Цифровой ввод-вывод
  clrf anselh        ; Цифровой ввод-вывод
  bcf status, 6      ; Назад на страницу 0
  clrf porta         ; Очистка всех портов
  clrf portb
  clrf portc
```

Порт A работает на ввод по умолчанию, а порт B необходимо сконфигурировать на вывод. Для считывания состояния переключателя выбора программы (S3) разряды RC0 и RC1 порта C должны быть входами. Остальные разряды должны быть выходами.

По умолчанию порт C принимает аналоговые входные или выходные сигналы, нам же требуется, чтобы он работал только в цифровом режиме. В микроконтроллере 16F690 это достигается обнулением разрядов в регистрах "аналогового выбора" (ANSEL и ANSELH). Это реализуют две команды `clrf`. Подпрограмма инициализации завершается очисткой всех трех регистров.

Если предполагается реализовать только один из названных режимов, то смело приступайте к вводу листинга. Если же необходимо использовать все четыре режима, то потребуется подпрограмма, считывающая состояние переключателя выбора программы на панели управления и направляющая микроконтроллер PIC к соответствующему фрагменту программы (листинг 9.2). Алгоритм этой подпрограммы демонстрирует блок-схема на рис. 9.33.

**Листинг 9.2. Подпрограмма выбора режима работы робота "Искатель"**

```
  btfsc portc, 1     ; Проверка разряда выбора режима 1
  goto bit1hi
  btfsc portc, 0     ; Проверка разряда выбора режима 0
  goto mode2         ; Пежим поиска света
```



**Листинг 9.2. Окончание**

```
  goto mode1         ; Режим "Скиталец"
bit1hi
  btfsc portc, 0     ; Проверка разряда выбора режима 0
  goto mode4         ; Режим "Узник"
  goto mode3         ; Режим отслеживания линии
```

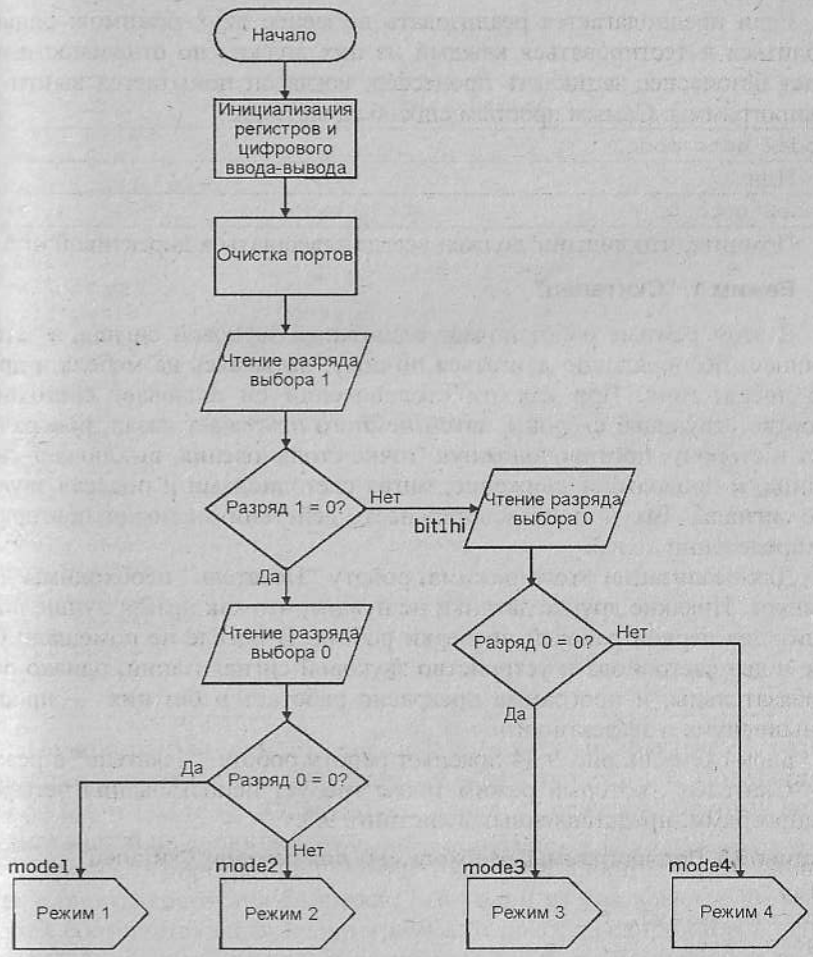



**Рис. 9.33.** Блок-схема подпрограммы выбора режима работы робота

Далее, состояние переключателя S4 проверяется опросом разряда RC1. Если он находится в состоянии лог. 0 (переключатель установлен в

позицию 0X), то программа продолжает работу проверкой состояния разряда 0. В зависимости от значения этого разряда программа переходит на режим 1 (00) или режим 2 (01). Единичное значение разряда (переключатель установлен в положение 1X) вызывает переход к метке `bit1hi`, где проверка состояния разряда 0 инициирует переход на режим 3 (10) или 4 (11).

Если предполагается реализовать не менее двух режимов, однако вводиться и тестироваться каждый из них должен по отдельности, то будет безопаснее зациклить процессор, когда он попытается выйти из подпрограммы. Самым простым способ сделать это:

```
mode2 goto mode2
```

Или:

```
mode2 goto $
```

Помните, что листинг должен всегда завершаться директивой `end`.

### Режим 1. "Скиталец"

В этом режиме робот подает одиночный звуковой сигнал, а затем начинает беспорядочно двигаться по полу, натыкаясь на мебель и другие препятствия. При каждом столкновении он включает светодиод с соответствующей стороны, затем немного отъезжает назад, поворачивает в сторону, противоположную точке столкновения, выключает светодиод, и продолжает движение, мигая светодиодами и подавая звуковые сигналы. Такую последовательность действий он может повторять неопределенно долго.

Для реализации этого режима, роботу "Искатель" необходимы два бампера. Никакие другие датчики не нужны, что как нельзя лучше подходит для первой рабочей проверки робота. В идеале не помешали бы еще и два светодиода и устройство звуковой сигнализации, однако они необязательны, и программа прекрасно работает и без них — просто меньше шума и эффектности.

Блок-схема на рис. 9.34 поясняет работу робота "Искатель" в режиме "Скиталец", который режим также требует использования четырех подпрограмм, представленных в листинге 9.3.

**Листинг 9.3. Подпрограммы, необходимые для режима "Скиталец"**

```
delay
  decfsz delay0, f
  goto delay
  decfsz delay1, f
  goto delay
  return
```



**Листинг 9.3. Окончание**

```
longdelay
  movwf delayn
repeat
  call delay
  decfsz delayn, f
  goto repeat
  return
avoidright
  bsf portc, 3      ; Включаем правый светодиод
  movlw 050h        ; Оба двигателя назад
  movwf portb
  movlw 0ah         ; Задержка = 10 x 0,2 = 2 с
  call longdelay
  movlw 090h        ; Поворот влево
  movwf portb
  movlw 0ah
  call longdelay
  clrf portb        ; Останов
  clrf portc        ; Выключаем светодиод
  return
avoidleft
  bsf portc, 4      ; Включаем левый светодиод
  movlw 050h
  movwf portb
  movlw 0ah
  call longdelay
  movlw 060h        ; Поворот вправо
  movwf portb
  movlw 0ah
  call longdelay
  clrf portb
  clrf portc
  return
```

Подпрограммы `avoidright` и `avoidleft` используются только в режиме 1, хотя вполне применимы и в других режимах. Подпрограммы `delay` и `longdelay` используются во всех режимах, поэтому не забудьте ввести их в программу.

Главная подпрограмма (листинг 9.4) — это цикл, который начинается с опроса состояния бамперов. Если один из них вдавлен, то вызывается соответствующая подпрограмма, и робот предпринимает действия по обходу препятствия, описанные ранее. Если бампер не задет, то робот движется вперед около 2 с, останавливается, подает звуковой сигнал и мигает светодиодами перед возвратом в начало цикла.

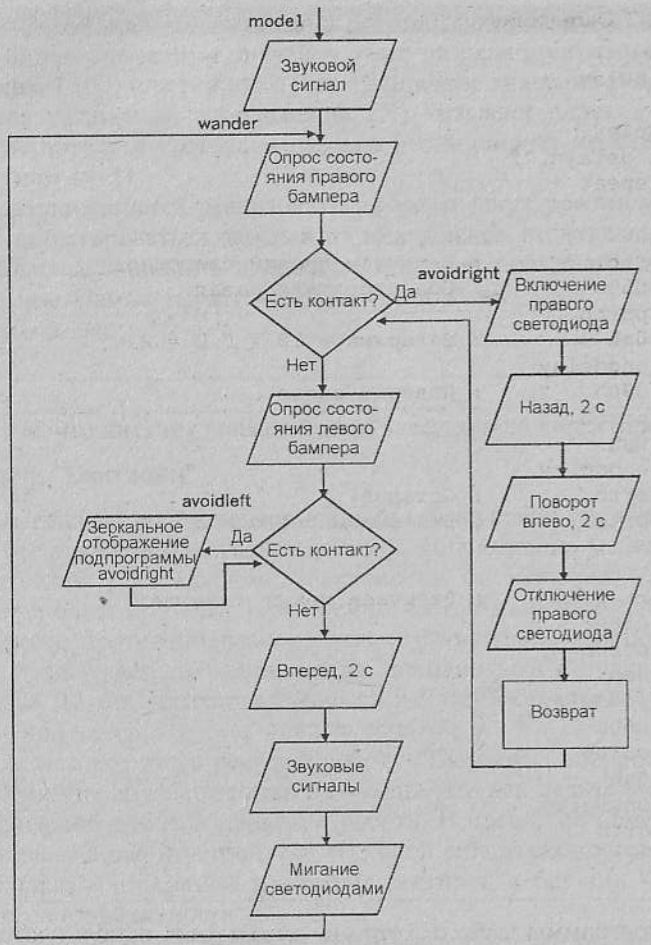



**Рис. 9.34.** Блок-схема режима 1

**Листинг 9.4. Главный цикл режима 1**

```
mode1
  bsf portc, 2      ; Звуковой сигнал, 0,2 с
  call delay
  bcf portc, 2
wander
  btfsc porta, 4    ; Опрос состояния правого бампера
  call avoidright
  btfsc porta, 5    ; Опрос состояния левого бампера
```



**Листинг 9.4. Окончание**

```
  call avoidleft
  movlw H'A0'       ; Код для 'прямо вперед'
  movwf portb       ; Оба двигателя вперед
  movlw H'0A'       ; Задержка 10 x 0,2 = 2 с
  call longdelay
  clrf portb        ; Останов
  bsf portc, 2      ; Короткий звуковой сигнал
  call delay
  bcf portc, 2
  call delay
  bsf portc, 2      ; Длинный звуковой сигнал
  movlw H'08'
  call longdelay
  bcf portc, 2
  bsf portc, 3      ; Мигание обоими светодиодами
  bsf portc, 4
  call delay
  clrf portc
  goto wander       ; Повтор без завершения программы
```

Подпрограмма `delay` вызывается несколько раз в цикле для обеспечения необходимого временного промежутка между повторными включениями и выключениями светодиодов и устройства звуковой сигнализации. Это формирует фиксированную задержку около 0,2 с.

Подпрограмма `longdelay` — более гибкая. Перед ее вызовом в рабочий регистр загружается шестнадцатеричное значение. Подпрограмма `longdelay` будет декрементировать это значение до тех пор, пока оно не станет нулевым. При этом на каждом шаге вызывается подпрограмма `delay`, в силу чего может формироваться задержка любой длительности вплоть до 50 с.

В подпрограмме `mode1` входные сигналы от бамперов опрашиваются с достаточно малыми интервалами. В микроконтроллере 16F690 входы порта A могут быть сконфигурированы на инициализацию прерывания по изменению состояния на входе. Таким образом, можно было бы использовать подпрограмму обслуживания прерываний, вызывающую подпрограмму обхода препятствия. Однако это усложняет программу, и поскольку нет никаких проблем с тем, чтобы робот двигался короткими шагами, проще использовать подпрограмму опроса.

### Режим 2. Поиск света

Эта программа лучше всего работает, когда робот находится в зашторенной комнате или плохо освещенном помещении. Она наиболее

эффективна в том случае, когда в комнате присутствует только один источник света. На послу не должно быть каких-либо препятствий. Роботу необходим фотоэлемент и пара бамперов.

Робот "Искатель" определяет направление на источник света в процессе поворота на месте с непрерывным опросом сигнала от фотоэлемента. Трубка, надетая на фотоэлемент, ограничивает угол обзора примерно десятью градусами. Как только робот распознает наличие перед ним света, он прекращает поворачиваться, входит в режим прямолинейного движения и начинает двигаться к источнику света, перемещаясь вперед в течении около трех секунд. Затем он останавливается и опрашивает состояние бамперов.

Если бамперы сигнализируют о том, что робот столкнулся с препятствием, то, скорее всего, достигнута лампа или окно. Робот немного отъезжает назад, выключает светодиод и программа на этом завершается. Если же бамперы не сигнализируют о наличии контакта, то опять считывается сигнал с фотоэлемента, и, если свет все еще виден, робот возобновляет движение вперед. Если свет не виден, то робот переходит в начало программы и опять начинает искать свет.

Общее количество сканирований ограничено числом 128. При этом сканированием считается один поворот влево длительностью 0,2 с. Всякий раз после потери источника света роботу может понадобиться несколько сканирований. Это число сохраняется в регистре, помеченном в начале программы как `scans`. Всякий раз, когда робот теряет направление, `scans` может декрементироваться несколько раз. Если после 128 отсчетов источник света не обнаружен, это может означать или то, что он заслонен каким-то крупным объектом, или что лампа была выключена. В этом случае робот переходит к выполнению подпрограммы `distress`, в ходе чего он выключает светодиоды и остается неподвижным, взывая о помощи путем подачи звуковых сигналов.

Вместо остановки и подачи звуковых сигналов робота можно было бы запрограммировать на выход в новую исходную точку с повторным поиском источника света в надежде обнаружить его из другой точки помещения. Эту процедуру можно повторить несколько раз вплоть до момента обнаружения света.

Обратите внимание, что в этом режиме вызываются подпрограммы `delay` и `longdelay` (см. листинг 9.3).

Блок-схема алгоритма для режима поиска света представлена на рис. 9.35, а соответствующая подпрограмма — в листинге 9.5.



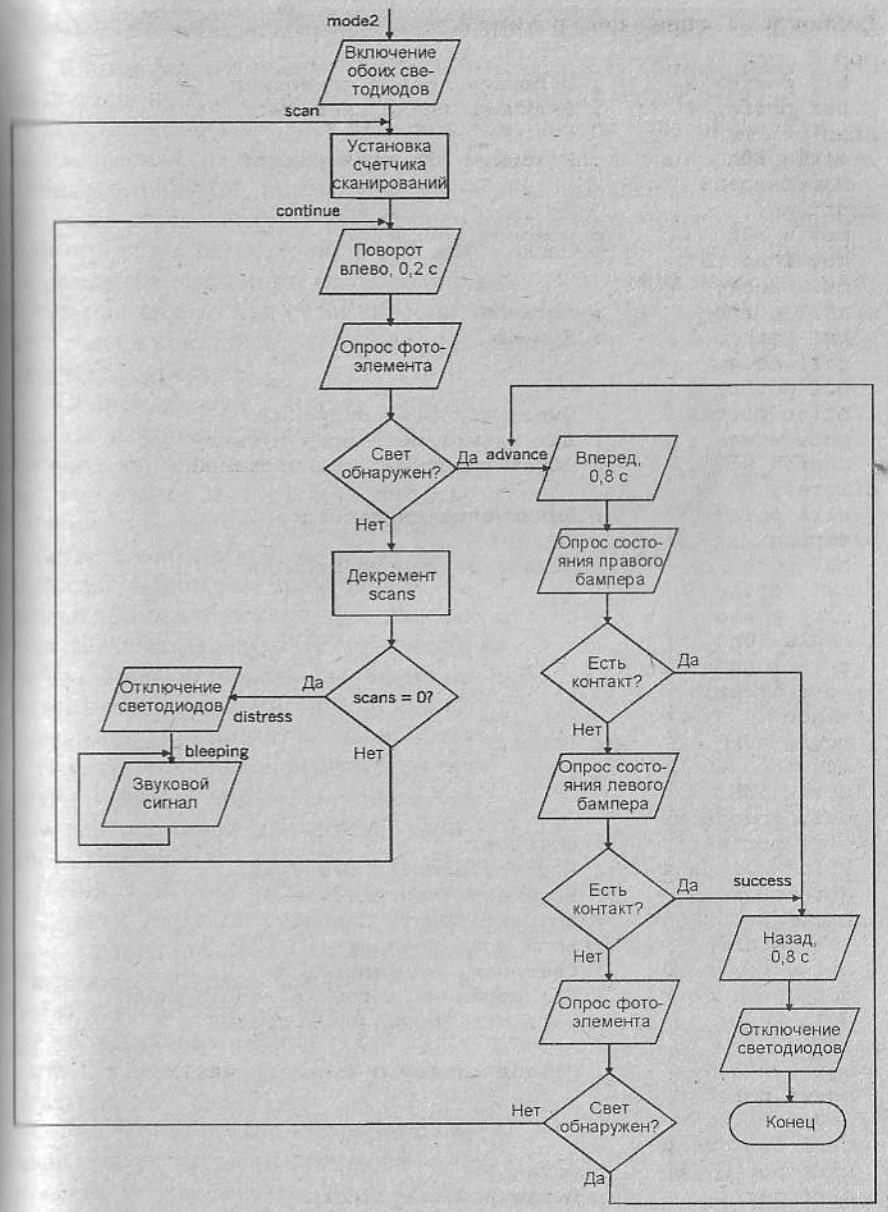



**Рис. 9.35.** Блок-схема для режима поиска света

**Листинг 9.5. Главный цикл режима 2**

```
mode2
  bsf portc, 3      ; Включаем правый светодиод
  bsf portc, 4      ; Включаем левый светодиод
scan
  movlw 80h         ; Максимум 128 сканирований
  movwf scans
continue
  movlw 90h         ; Поворот влево
  movwf portb
  call delay
  clrf portb        ; Останов
  bsf portc, 2      ; Зуммер
  call delay
  bcf portc, 2
  btfsc porta, 0    ; Свет распознан впереди?
  goto advance      ; Да - движение вперед к свету
  decfsz scans, f   ; Обратный отсчет сканирований
distress
  clrf portc        ; Выключение светодиодов
bleeping
  bsf portc, 2      ; Подача звукового сигнала
  call delay
  clrf portc
  movlw 10h
  call longdelay
  goto bleeping
advance
  movlw a0h         ; Вперед
  movwf portb
  movlw 10h
  call longdelay
  clrf portb        ; Останов
  btfsc porta, 4    ; Контакт правого бампера?
  goto success      ; Если есть контакт
  btfsc porta, 5    ; Контакт левого бампера?
  goto success      ; Если есть контакт
  btfsc porta, 0    ; Свет все еще виден?
  goto advance      ; Да, движение в том же направлении
  goto scan         ; Нет, повторный поиск света
success
  movlw H'50'       ; Отъезд от лампы (она горячая!)
  movwf portb
  movlw H'10'
  call longdelay
  clrf portb        ; Останов
  clrf portc        ; Выключаем светодиоды
  goto $            ; Ожидание
```



### Режим 3. Отслеживание линии

В этом режиме робот "Искатель" отслеживает линию, нанесенную на пол или на другую поверхность. Для решения этой задачи ему необходима пара инфракрасных датчиков. Светодиоды высокой яркости, устанавливаемые на передней панели робота, не являются критически-важным атрибутом, однако они улучшают визуальное восприятие — особенно в условиях слабой освещенности. Робот может, конечно же, работать как в темноте, так и при ярком освещении. Другой практический момент, связанный со светодиодами: когда робот поворачивается влево или вправо при отслеживании траектории, со стороны поворота включается светодиод. Это помогает убедиться в том, что программа работает корректно.

Отслеживаемую линию можно сделать из липкой черной ленты или черного картона или же нарисовать краской. Перед этим следует убедиться в том, что используемый материал действительно поглощает инфракрасный свет. Некоторые виды черной бумаги и картона отражают лучи из этой области спектра, поэтому черная линия из такого материала для робота не будет отличаться от белой или слегка окрашенной поверхности. Мы пришли к выводу, что приемлемый результат дает рисование линии на тонком белом картоне серой или черной акриловой краской.

Линия должна быть около 20 мм в ширину. Нарисуйте замкнутую петлю с прямолинейными и изогнутыми участками. Радиус каждой кривой должен составлять не менее 70 мм. При более резких изгибах робот будет пытаться пересечь линию и останавливаться.

Рис. 9.36 поясняет, как робот с помощью двух датчиков определяет отражение инфракрасного света. Обычно, когда робот перемещается вдоль

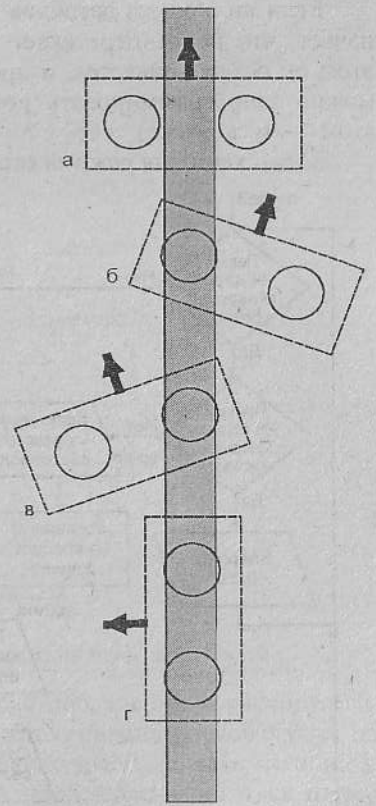



**Рис. 9.36.** Четыре положения датчиков относительно линии

прямой, датчики позиционируются по обе стороны от линии (положение *а*). Робот перемещается вперед в течении 0,2 с и повторно опрашивает состояние датчиков.

Как только один из датчиков воспринимает белый цвет, это указывает на поворот траектории. Реакция робота зависит от того, какой из датчиков видит белый цвет, а какой — черный. Так положению *б* на рис. 9.36 соответствует поворот влево. Для того чтобы остаться на линии, робот должен повернуться на месте влево. Программа переходит к метке `spinl`, включается левый светодиод и в рабочий регистр загружается значение `H'90'`, и двигатель отрабатывает управляющий код для поворота влево. Противоположное происходит, если робот должен повернуть вправо (положение *в* на рис. 9.36).

Если ни один из датчиков не воспринимает белого цвета, то это означает, что робот пересекает линию (положение *г* на рис. 9.36). При этом он останавливается, и программа завершается. Для такого случая можно запрограммировать робота делать что-нибудь интересное (об этом — чуть позже).

Блок-схема для режима отслеживания линии показана на рис. 9.37.

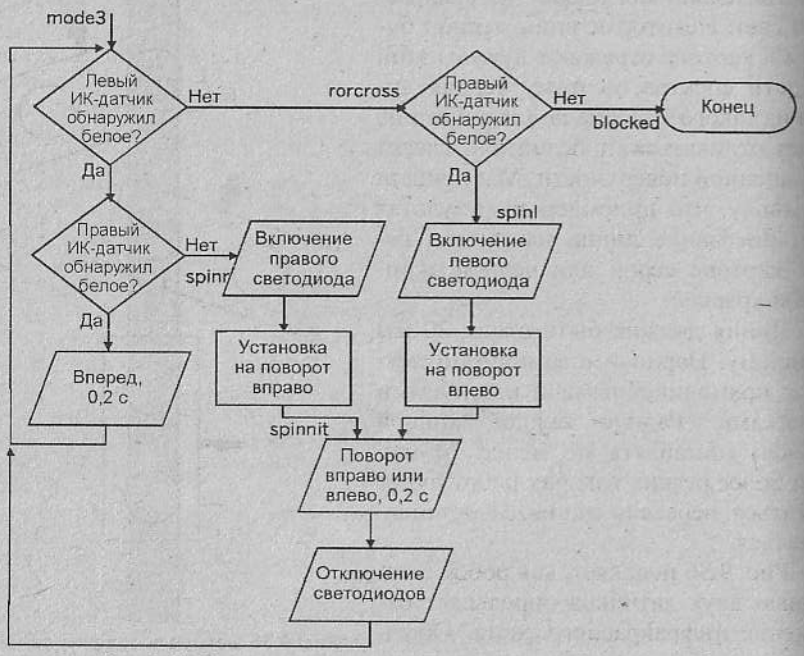



**Рис. 9.37.** Блок-схема для подпрограммы отслеживания линии



В случаях *б* и *в* программа переходит к метке `spinl` или `spinr` в зависимости от того, в каком направлении должен повернуться робот. Рабочий регистр `w` загружается соответствующим кодом, и программа переходит к метке `spinnit`. Значение, сохраненное в регистре `w`, используется для поворота робота в требуемом направлении. Программа повторяется, и робот следует по замкнутой траектории неопределенно долго до момента блокировки движения при попытке пересечь линию.

Подпрограмма отслеживания линии показана в листинге 9.6.

**Листинг 9.6. Главный цикл режима 3**

```
mode3
  btfsc porta, 2     ; Левый инфракрасный датчик
  goto rorcross      ; Уходит вправо или пересекает кривую
  btfsc porta, 1     ; Правый инфракрасный датчик
  goto spinr         ; Уходит влево
  movlw a0h          ; Вперед
  movwf portb
  call delay
  clrf portb         ; Останов
  goto mode3         ; Бесконечный цикл
rorcross
  btfsc porta, 1     ; Правый датчик
  goto blocked       ; Пересечение
  goto spinl         ; Уходит вправо
spinr
  bsf portc, 3       ; Включаем правый светодиод
  movlw 60h          ; Поворот вправо
  goto spinnit
spinl
  bsf portc, 4       ; Включаем левый светодиод
  movlw 90h          ; Поворот влево
spinnit
  movwf portb        ; Поворот
  call delay
  clrf portb         ; Останов
  clrf portc         ; Выключаем светодиод
  goto mode3
blocked
  goto $             ; Ожидание (или дальнейшие действия)
```

В эту программу можно внести много добавлений и расширений. Например, запрограммируйте робота на отслеживание белой линии на темной поверхности. Можно также поэкспериментировать с линией. Если робот подходит к ее концу, то он продолжает двигаться прямо вперед некоторое короткое расстояние, после чего может распознать начало второй линии и приступить к ее отслеживанию. Используйте

этот вариант поведения для реализации следования робота по траектории в виде восьмерки.

Робот может также проходить лабиринт из разветвляющихся линий. В каждой точке ветвления он останавливается и случайным образом выбирает направление движения вправо или влево. В конце концов робот приходит в одну из нескольких конечных точек, из которых только одна является "выходом". Робот должен запоминать свой выбор в каждой точке ветвления и тем самым учиться находить путь к правильной конечной точке.

Программа прохождения подобного лабиринта представлена в листинге 9.7, а соответствующий ей HEX-файл — в листинге 9.8.

Соответствующие файлы `Quest05.asm` и `Quest05.hex` можно также найти на прилагаемом к книге компакт-диске в папке `Искатель`.

**Листинг 9.7. Файл Quest05.asm**

```
;*****************************************************
;
;  Имя файла: Quest05.asm
;  Прохождение лабиринта из линий
;
;*****************************************************

  list       p=16F690
  __config   0x33c4

  status     equ 03h
  porta      equ 05h
  portb      equ 06h
  portc      equ 07h
  trisa      equ 05h
  trisb      equ 06h
  trisc      equ 07h
  ansel      equ 1Eh
  anselh     equ 1Fh
  w          equ 00h
  f          equ 01h
  z          equ 02h
  delay0     equ 20h
  delay1     equ 21h
  delayn     equ 22h
  scans      equ 23h
  randval    equ 24h
  bitn       equ 25h
  bitm       equ 26h
```



**Листинг 9.7. Продолжение**

```
  junction   equ 27h
  mask       equ 28h
  route      equ 29h
  resultflag equ 2ah
  copyroute  equ 2bh

; Программа начинается здесь

  goto start
  org 0004h
  goto start

start
  bsf status, 5      ; Банк 1
  clrf trisb         ; Все порты B - выходы
  movlw 03h          ; RC0 и RC1 - входы
  movwf trisc
  bcf status, 5      ; Возвращаемся к банку 0
  bsf status, 6      ; Банк 2
  clrf ansel         ; Цифровой ввод-вывод
  clrf anselh        ; Цифровой ввод-вывод
  bcf status, 6      ; Возвращаемся к банку 0
  clrf porta         ; Обнуляем все порты
  clrf portb
  clrf portc
  clrf resultflag
  bsf randval, 0     ; Устанавливаем основу для случайных чисел
ticking
  call flashem
  call randno
  btfsc porta, 3     ; Кнопка нажата?
  goto ticking
nopress1
  btfss porta, 3     ; Кнопка отпущена?
  goto nopress1
newrun
  movlw 04h          ; Устанавливаем счетчик пересечений
  movwf junction
  btfsc resultflag, 0      ; Старый маршрут правильный?
  goto setroute      ; Продолжаем следовать по старому маршруту
  call randno        ; Новые разряды <2:0>
  call randno
  call randno
  movlw 07h          ; Устанавливаем 3 младших разряда
  andwf randval, w   ; Выбираем младшие 3 разряда randval
  movwf route        ; Сохраняем как новый маршрут
```

**Листинг 9.7. Продолжение**

```
setroute
  movf route, w
  movwf copyroute   ; Копируем старый или новый маршрут
follow
  movf junction, w  ; Начинаем идти к следующему пересечению
  btfsc status, z   ; z = 1?
  goto learnit      ; Конечный пункт
  btfsc porta, 2    ; Левый ИК-датчик
  goto rorjunct     ; Уход вправо или на пересечении
  btfsc porta, 1    ; Правый ИК-датчик
  goto spinr        ; Уход влево
  movlw 0a0h        ; Вперед
  movwf portb
  call delay
  clrf portb        ; Останов
  goto follow       ; Повторяем бесконечно
rorjunct
  btfsc porta, 1    ; Правый датчик
  goto branch       ; Подходим к пересечению
  goto spinl        ; Уход вправо
spinr
  bsf portc, 3      ; Включаем правый светодиод
  movlw 60h         ; Поворот вправо
  goto spinnit
spinl
  bsf portc, 4      ; Включаем левый светодиод
  movlw 90h         ; Поворот влево
spinnit
  movwf portb       ; Фактический поворот
  call delay
  clrf portb        ; Останов
  clrf portc        ; Отключаем светодиод
  goto follow
branch
  decfsz junction,f ; В конечном пункте?
  goto whichway     ; Поворот влево или вправо?
  goto learnit;     ; Конечный пункт
whichway
  rrf copyroute, f  ; Младший разряд - в флаг переноса
  btfss status, 0   ; Опрос флага переноса
  goto turnl        ; Перенос = 0, поэтому поворачиваем влево
  goto turnr        ; Перенос = 1, поэтому поворачиваем вправо
turnl
  bsf portc, 2      ; Зуммер
  movlw 90h         ; Поворот влево, 35 градусов
  movwf portb
```



**Листинг 9.7. Продолжение**

```
  bsf portc, 4      ; Включаем левый светодиод
  movlw 06h
  call longdelay
  bcf portc, 4      ; Выключаем светодиод
  bcf portc, 2      ; Выключаем зуммер
  goto crossit      ; Для пересечения маркера
turnr
  bsf portc, 2
  movlw 60h         ; Поворот вправо, 35 градусов
  movwf portb
  bsf portc, 3      ; Включаем правый светодиод
  movlw 06h
  call longdelay
  bcf portc, 3      ; Выключаем светодиод
  bcf portc, 2
crossit
  movlw 0a0h        ; Вперед
  movwf portb
  movlw 0ah
  call longdelay    ; Начинаем движение по новой ветви
  clrf portb        ; Останов
  goto follow
learnit
  btfsc porta, 4    ; Правый бампер нажат?
  goto success
  btfsc porta, 5    ; Левый бампер нажат?
  goto fail
  goto learnit      ; Ожидаем нажатия бампера
success
  bsf resultflag, 0 ; 1 = правильный конечный пункт
  goto nextgo
fail
  bcf resultflag, 0 ; 0 = неверный конечный пункт
nextgo
  bsf portc, 2      ; Включаем зуммер
  movlw 0a0h        ; Вперед
  movwf portb
runfree
  btfsc porta, 1    ; Опрашиваем правый датчик
  goto runfree      ; Пока не исследует конечный пункт
  call shortdelay
  bcf portc, 2      ; Выключаем зуммер
  movlw 0a0h        ; Вперед
  movwf portb
resetline
  btfss porta, 1    ; Опрашиваем правый датчик
```

**Листинг 9.7. Продолжение**

```
  goto meetline      ; Пока не пересечется с линией
straddleline
  btfss porta, 2     ; Опрашиваем левый датчик
  goto straddleline  ; Пока не пересечет линию
  movlw 90h          ; Поворот влево
  movwf portb
  call flashem
  call flashem
  goto newrun

 ; Подпрограммы

delay
  decfsz delay0, f
  goto delay
  decfsz delay1, f
  goto delay
  return
longdelay
  movwf delayn
repeat
  call delay
  decfsz delayn, f
  goto repeat
  return
shortdelay
  movlw 08h
  movwf delay1
  call delay
  clrf delay1
  return
randno
  clrf bitn
  clrf bitm
  btfsc randval, 5   ; Получаем n
  bsf bitn, 0        ; Если n = 1
  btfsc randval, 6   ; Получаем m
  bsf bitm, 0        ; Если m = 1
  movf bitn, w       ; n в w
  xorwf bitm, w      ; XOR m и n, результат - в w
  addlw 0ffh         ; Устанавливаем флаг переноса, если w = 1
  rlf randval,f      ; Новое случайное число в randval
  return
flashem
  bsf portc, 3
  bsf portc, 4
```



**Листинг 9.7. Окончание**

```
  call delay
  bcf portc, 3
  bcf portc, 4
  call delay
  return

end
```

**Листинг 9.8. Файл Quest05.hex**

```
:020000040000FA
:020000000528D1
:0800080005288316860103307O
:10001000870083120317 9E019F0103138501860148
:10002000870 1AA012414902085208519132885 1D95
:1000300017280430A7002A18232885208520852O2A
:10004000073024O5A9002908AB00270803195C28FC
:10005000051931288518342 8A030860076208601BD
:10006000252885183E28372887156030392807 1637
:10007000903086007620860187012528A70B41282D
:100080005C28AB0C031C45284E280715903086 00D1
:10009000071606307B2007120711562807156030 17
:1000A000860871506307B2087110711A030860057
:1000B0000A307B2086012528051A6128851A6328C5
:1000C0005C282A1464282A100715A0308600851899
:1000D000672880200711A0308600851C6D28051D2B
:1000E0006F2890308600902090201928A00B762849
:1000F000A10B76280800A2007620A20B7C2808001D
:10010000083 0A1007620A1010800A501A601A41ACB
:10011000251424 1B261425082606FF3EA40D0800DE
:0E0120008715071676208711071276200800 33
:02400E00C433B9
:00000001FF
```

### Режим 4. "Узник"

Этот режим требует пары инфракрасных датчиков и дополнительно — пары светодиодов и зуммера. Робот "Искатель" помещается в область, окруженную непрерывной линией, из которой он не может "сбежать", поскольку запрограммирован не пересекать линию. В этой двухмерной "тюрьме" робот движется хаотически. При желании можете оставить в линии узкий разрыв, чтобы увидеть, сколько времени понадобится "Искателю" для того, чтобы найти выход и известить об этом подачей триумфального звукового сигнала.

Блок-схема этого режима показана на рис. 9.38.

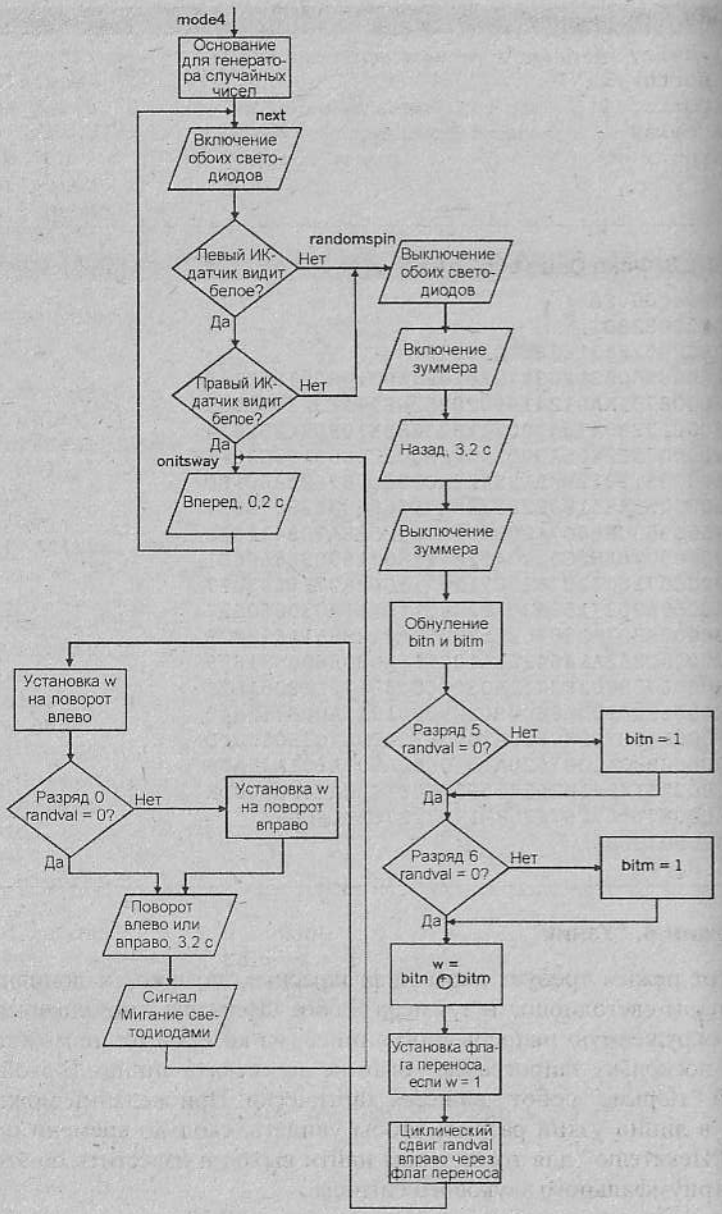



**Рис. 9.38.** Блок-схема для подпрограммы "Узник"



Подпрограмма начинается с простого цикла, в котором робот движется вперед с включенными передними светодиодами до тех пор, пока какой-либо из датчиков не распознает черную линию. В результате программа перейдет к подпрограмме `randomspin`, которая выключает светодиоды и включает зуммер, подающий звуковые сигналы в то время, как робот разворачивается в течение около 3,2 с. Затем генерируется случайное число (0 или 1), которое сохраняется в переменной `randval`. После этого код, передаваемый двигателям, первоначально устанавливается в `90h` (поворот влево), однако он изменяется на `60h` (поворот вправо), если сгенерировано число 1. После поворота в случайно выбранном направлении и нескольких миганий и звуковых сигналов робот возобновляет движение вперед. Эта подпрограмма работает неопределенно долго (листинг 9.9).

**Листинг 9.9. Главный цикл режима 4**

```
mode4
  movlw 33h
  movwf randval      ; Основание для генератора случайных чисел
                     ;  в randval
next
  bsf portc, 3       ; Включаем оба светодиода
  bsf portc, 4
  btfsc porta, 2     ; Левый инфракрасный датчик
  goto randomspin    ; Блокировка для левого направления
  btfsc porta, 1     ; Правый инфракрасный датчик
  goto randomspin    ; Блокировка для правого направления
onitsway
  movlw a0h          ; Вперед
  movwf portb
  call delay
  clrf portb         ; Останов
  goto next          ; Бесконечное повторение
randomspin
  clrf portc         ; Выключаем светодиоды
  bsf portc, 2       ; Включаем зуммер
  movlw 50h          ; Назад
  movwf portb
  movlw 10h
  call longdelay
  clrf portc         ; Выключаем зуммер
  clrf portb         ; Останов
  clrf bitn
  clrf bitm
  btfsc randval, 5   ; Получаем n
  bsf bitn, 0        ; Если n = 1
```

**Листинг 9.9. Окончание**

```
btfsc randval, 6  ; Получаем m
bsf bitm, 0       ; Если m = 1
movf bitn, w      ; n в w
xorwf bitm, w     ; Исключающее ИЛИ m и n, результат - в w
addlw 0ffh        ; Установка флага переноса, если w = 1
rlf randval,f     ; Новое случайное число в randval
movlw 90h         ; Установка левого поворота
btfsc randval, 0  ; Получение разряда 0 случайного числа
movlw 60h         ; Изменение установки на поворот вправо
movwf portb       ; Поворот влево или вправо
movlw 10h
call longdelay
clrf portb        ; Останов
bsf portc, 2      ; Звуковой сигнал
bsf portc, 3      ; Мигание светодиодами
bsf portc, 4
call delay
clrf portc
goto onitsway
```

## Разработка робота "Искатель"

Рассмотренные подпрограммы можно по-разному объединять вну ри одной более сложной программы — благо микроконтроллер PI 16F690 предоставляет в распоряжение достаточный объем памяти пр грамм.

Можно встроить в этот робот какие-либо другие датчики, рассмо ренные в главе 3. Так, например, звуковой датчик применим в само широком диапазоне: от реакции на хлопок в ладоши (при этом в поме щении должно быть очень тихо) до обеспечения информационного об мена между парой роботов с помощью звуковых сигналов. Ультразву ковой передатчик и приемник обеспечивают еще один способ распозна вания и определения местоположения препятствий.

Робот "Искатель" оснащен очень чувствительным направленны фотоэлементом, который можно использован как в программах поиск света, так и в программах удаления от источника освещения.

Что касается конструкции робота, то при желании скрыть его на чинку, придумайте для него "кожу". Используйте для этого лист ПВХ пенопласт или толстый картон. "Кожа" может скрывая печатные плат батареи и провода.

Различные примеры программ для робота "Искатель" можно найти на прилага мом к книге компакт-диске в папке Искатель.

# Глава 10. Портальный робот

Портальные роботы чаще встречаются на промышленных предпри тиях, чем в мастерских любителей робототехники. Они обычно состоят из главной рамы, которая поддерживает пару направляющих (так назы ваемых **Y-направляющих**), по которых ездит снабженная колесами **Y-рама**. Эта рама несет вторую пару направляющих (**X-направляющие**), по которым ездит снабженная колесами **X-рама** (рис. 10.1).

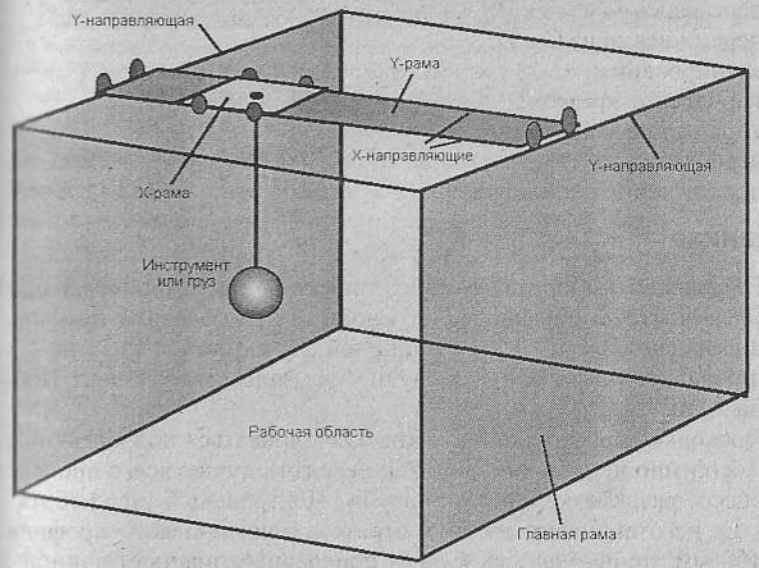



**Рис. 10.1.** Структура типичного портального робота

X-рама несет инструмент или груз, который может подниматься и опускаться. В результате перемещения рам в направлениях X и Y, а также подъема и опускания инструмент может быть точно размещен над любой точкой рабочей области.

В случае приложений наподобие настольных игр портальный робот имеет то преимущество, что положение его манипулятора известно с большой точностью. Этого не скажешь об обычных мобильных робо-

тах, в которых методики локализации обычно подвержены накапливающимся погрешностям.

Спецификация портального робота:

- конструкция из алюминиевого профиля;
- главная рама плюс рамы X и Y;
- два двигателя для рам X и Y;
- один двигатель для подъема и опускания инструмента;
- микроконтроллер PIC 16F690;
- два инфракрасных датчика или датчика на основе эффекта Холла для определения позиций X и Y;
- инструменты: крюк, захват, кисть для рисования, лазерный луч, камера.

Программы:

- перемещение из A в B;
- перемещение из C в D;
- сканирование;
- управление крюком;
- управление захватом;
- игра "Подкова";
- прохождение лабиринта.

## Механика

Идеальный выбор для данного проекта — алюминиевый профиль. В нашем опытном образце мы в основном использовали профиль с П-образным сечением 15×15 мм толщиной 3 мм. Для панелей, на которых монтировались электронные модули, был задействован лист ПВХ толщиной 3 мм.

Поскольку конструкция должна адаптироваться под имеющиеся детали (особенно двигатели и коробки передач), лучше всего продвигаться в процессе разработки изнутри наружу. Центральный узел робота — X-рама. Ее изготавливают из двух отрезков алюминиевого профиля длиной 100 мм, привинченных к двум поперечным планкам длиной 50 мм (рис. 10.2).

Элементы X-рамы, как и большая часть узлов портального робота, скреплены болтами M3 и гайками с виброустойчивыми шайбами. Гайки зафиксированы на резьбе с помощью анаэробного клея.

В центре каждой поперечной планки X-рамы сверлится отверстие, предназначенное для привязывания троса, тянущего X-раму вдоль направляющих. В длинных отрезках сверлятся отверстия, предназначенные для установки колес и ПВХ-панели, имеющей габариты 70×50 мм.



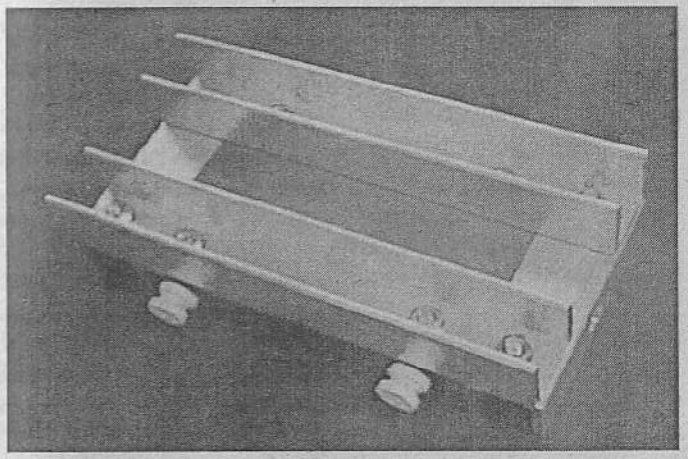


**Рис. 10.2.** X-рама собрана из четырех отрезков алюминиевого профиля, скрепленных болтами. Прямоугольная форма узла поддерживается ПВХ-панелью, привинченной болтами к длинным отрезкам профиля. Эти же болты используются для фиксации колесных баз

Для данного робота лучше всего подходят колеса из конструктора Lego™ (рис. 10.3).


**Рис. 10.3.** Базовый узел (черный) с одним установленным колесом (белое). Канавка колеса достаточно широкая, чтобы плотно держаться на направляющей шириной 3 мм

Мы использовали меньший из имеющихся двух размеров колес (приблизительно 8 мм в диаметре). Ступицы колес (без шин) плотно надеваются на выступы квадратных баз, обеспечивая минимум трения, что для данного случая — идеально. Колесная база имеет два выступа для установки колес, из которых неиспользуемый спиливается. В центре базы сверлится отверстие диаметром 3 мм, предназначенное для привинчивания узла к раме.

Когда рама собрана (рис. 10.4), расстояние между канавками на противоположных колесах составляет 57 мм. Обязательно выясните этот размер перед продолжением сборочных работ, поскольку от него зависят размеры следующего звена: Y-рамы.

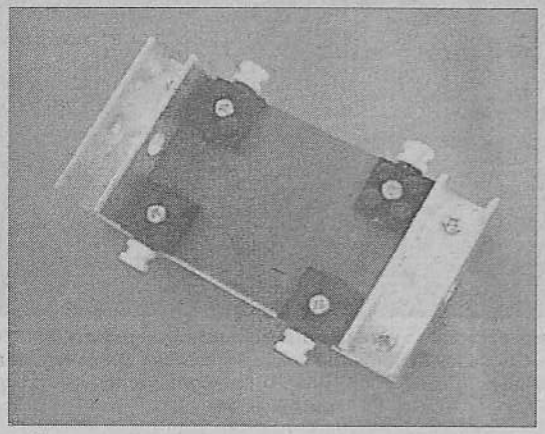


**Рис. 10.4.** Вид на X-раму снизу, демонстрирующий установку четырех колес. На этом этапе какие-либо средства крепления инструмента отсутствуют

Габариты описанной выше X-рамы минимальны для удержания размеров завершенного портального робота в приемлемых границах, однако никто не мешает сделать ее шире и/или длиннее для возможности установки конкретных инструментов.

Y-рама (рис. 10.5) значительно длиннее X-рамы, поскольку должна обеспечивать перемещение инструмента по всей ширине рабочей области. В данном случае мы предполагаем, что рабочая область будет похожа на шахматную доску с 64 квадратами, сгруппированными в восьми строках и восьми столбцах. Если сторона каждого квадрата составляет 30 мм, то минимальная длина Y-рамы равна

ширина поперечины × 2 + длина панели двигателя +
длина X-рамы + ширина рабочей области +
свободное пространство по обе стороны рабочей области.

В опытном образце соответствующие размеры были следующими: (15 × 2 ) + 80 + 100 + (30 × 8) + 40 = 450 мм.

Как показано на рис. 10.6, поперечины выступают за края рамы. Колесные базы — квадратные со стороной около 16 мм, поэтому поперечины должны выступать за раму на 17 мм с каждой стороны. Общая длина поперечины определяется как:



промежуток между X-направляющими + ширина профиля × 2 + выступ × 2.

В опытном образце соответствующие размеры имели значения: 57 + (15 × 2) + (17 × 2 ) = 121 мм.

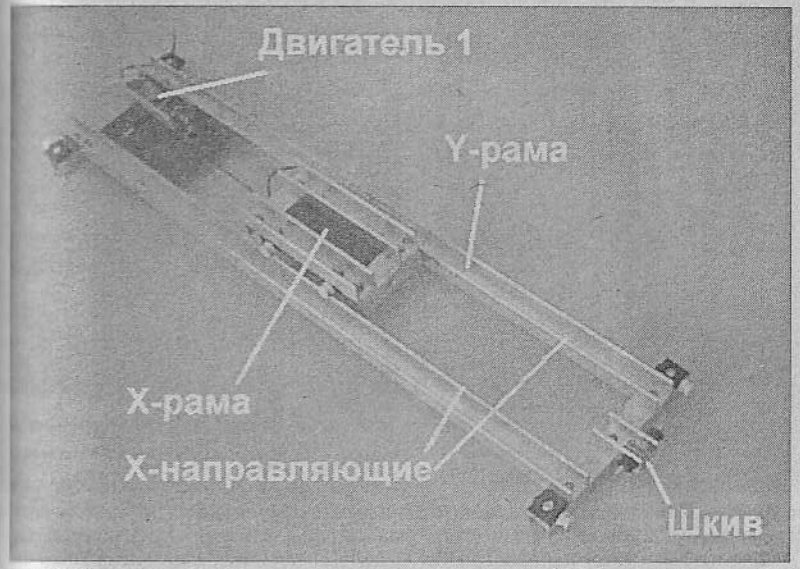



**Рис. 10.5.** Завершенная Y-рама с двигателем перемещения X-рамы, установленным на панели, расположенной у левого края Y-рамы. Шкив для троса противовеса установлен на правом крае рамы

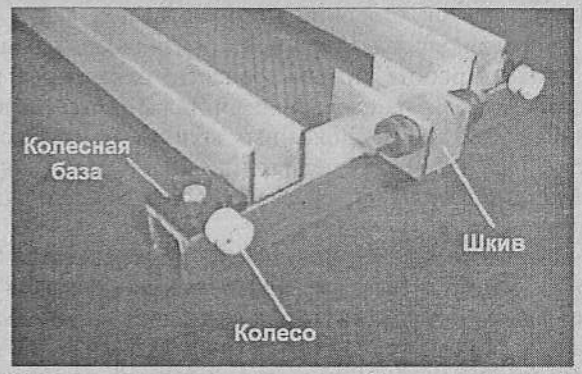



**Рис. 10.6.** Край Y-рамы, на котором установлен шкив. Узел шкива должен быть вынесен по крайней мере на 20 мм за край рамы, чтобы ход троса перекрывал всю длину Y-рамы

Главная рама состоит из передней, задней и двух боковых направляющих, скрепленных по углам болтами (рис. 10.7).


**Рис. 10.7.** Завершенная главная рама. Y-рама перемещается между передней и задней поперечинами по боковым направляющим главной рамы. Передняя панель используется для монтажа микроконтроллера PIC и других электронных схем управления

Эти направляющие вырезаны из алюминиевого П-образного профиля и скреплены болтами. При этом передняя и задняя направляющие ориентированы открытой стороной профиля вверх, а две боковые — вниз. Длина направляющих выбирается произвольно, однако помните, что рабочая область робота меньше области, охваченная рамой. Расстояние между боковыми направляющими должно быть равно расстоянию между канавками противоположных колес Y-рамы. В опытном образце это составляет 455 мм. Передняя и задняя направляющие — длиной 485 мм (при ширине профиля 15 мм), а боковые направляющие — 510 мм.

Рама включает в себя переднюю панель размерами 485×100 мм, изготовленную из ПВХ. В ее углах просверлены отверстия диаметром 3 мм для прикрепления болтами *между* передней и боковыми направляющими главной рамы. Это означает, что на болтах между задней и боковыми направляющими на задних углах главной рамы должны быть установлены шайбы или гайки эквивалентной толщины.



Учтите также, что еще одной функцией передней панели является поддержка прямоугольной формы рамы.

Сборка портального робота завершается установкой четырех стоек, вырезанных из алюминиевого П-образного профиля. Их длина выбирается произвольно (так, в опытном образце она составила 340 мм). Стойки должны быть достаточно длинными для того, чтобы противовесы могли перемещать обе рамы на всю длину направляющих. Передние две стойки привинчиваются болтами к боковым направляющим сразу же за передней панелью. При сверлении отверстия у верхнего края стойки просверлите *обе* стороны профиля, чтобы обеспечить доступ для отвертки (рис. 10.8).


**Рис. 10.8.** Передняя правая стойка, привинченная болтом к правой направляющей. Обратите внимание на отверстие, предназначенное для обеспечения доступа отвертки

Сборка задних стоек немного сложнее, поскольку они располагаются на задних углах главной рамы и должны привинчиваться болтами как к боковой, так и к задней направляющей. При разметке стоек для сверления следует учитывать наличие шайбы или гайки между направляющими (рис. 10.9).

Завершенный портальный робот показан на рис. 10.10. Следующий этап — установка лебедок и концевых микропереключателей. Вначале рассмотрим лебедки: их размещение и назначение. X-лебедка перемещает X-раму по X-направляющим. В качестве привода используется небольшой двигатель постоянного тока M1, работающий от напряжения в 3 В. Этот двигатель соединяется с X-рамой тросом, который может наматываться или разматываться в зависимости от направления враще-

ния вала (рис. 10.11). Трос поддерживается в натянутом состоянии противовесом, прикрепленным на другом конце рамы. Аналогичный метод используется и для перемещением Y-рамы.


**Рис. 10.9.** Правая задняя стойка, привинченная болтом к правой и задней направляющим (болт не виден). Обратите внимание на отверстие, предназначенное для доступа отвертки


**Рис. 10.10.** Завершенный портальный робот с установленными лебедками и некоторыми электронными блоками. Алюминиевая полоса, соединяющая две передних стойки, увеличивает жесткость конструкции робота



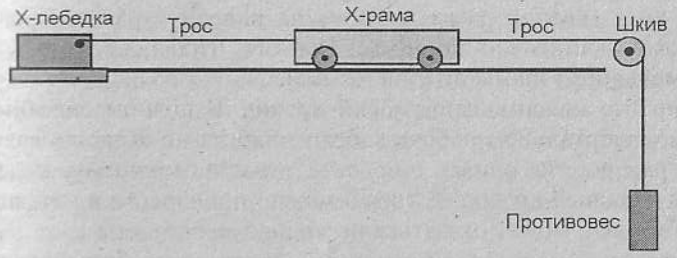



**Рис. 10.11.** Простой лебедочный механизм для перемещения оснащенной колесами рамы

Противовес должен быть достаточно тяжелым для того, чтобы толкать X-раму вдоль направляющих по мере раскручивания лебедки, однако не настолько тяжелым, чтобы двигатель не смог вновь намотать трос. В опытном образце в качестве противовеса использовались четыре рыболовных грузила по 35 грамм каждый, однако подошел бы и любой другой компактный предмет примерно такого же веса. Y-рама перемещается аналогичным механизмом. Его лебедка монтируется на передней панели главной рамы, а в качестве противовеса используется восемь грузил по 35 грамм.

### Двигатели

В проекте используются три двигателя постоянного тока, выполняющие роль лебедок (рис. 10.12). Двигатель 1 смонтирован в центре передней панели и перемещает Y-раму. Двигатель M2 монтируется с одной стороны Y-рамы и перемещает X-раму. Двигатель M3 монтируется на Y-раме и поднимает или опускает рабочий инструмент.

В этом проекте могут присутствовать еще один-два двигателя для управления инструментами наподобие схвата или кисти.

Питание и управление всеми тремя лебедками реализовано от платы питания двигателей, смонтированной на передней панели. Пары проводников идут к M2 и M3 от контактного блока, установленного


**Рис. 10.12.** Три лебедки. Двигатель 3 находится над панелью, а его лебедка расположена ниже. Двигатель 2 — напротив вверху. Лебедки 2 и 3 расположены у центральной оси Y-рамы во избежание возникновения боковых сил, стремящихся разрушить направляющие X-рамы. Лебедка 1 монтируется по центру передней панели

у левого края главной рамы, к блоку на панели Y-рамы. Проводни должны быть достаточно длинными для того, чтобы обеспечить свобо ное перемещение Y-рамы по главной раме.

Выбирайте максимально гибкий провод. В полномасштабном пр мышленном портальном роботе кабели никогда не ограничивают пер мещение рам робота, однако в проекте, подобном нашему, даже сам гибкий монтажный провод, который можно приобрести в магазинах д радиолюбителей, может оказаться несколько жестковатым.

Для проекта мы выбрали недорогие двигатели небольшого разме со встроенной коробкой передач (с фиксированным понижающим пер даточным числом) и далеко выступающим валом. При напряжении п тания 12 В выходной вал вращается со скоростью 36 об/мин, обеспеч вая выходной вращающий момент 12 кг*см. Двигатель может работа при любом напряжении из диапазона 4,5..18 В, выдавая пропорциона ные скорости и вращающие моменты. При этом было обнаружено, чт при намотке троса непосредственно на вал двигателя рама и инструме ты перемещаются слишком медленно.

Для повышения скорости движения рамы мы установили на валах приводных двигателей металлические катушки (вполне приемлемы катушки для швейных машинок) (рис. 10.13). Диаметр вала двигателя обычно составляет 2 или 4 мм, однако стандартная швейная катушка требует для своей установки вала диаметром 6 мм.

Для плотной посадки катушки на вал двигателя можно воспользоваться пластмассовым прутом. Втисните отрезок прута длиной 10 мм в центральное отверстие катушки, после чего просверлите вдоль его оси отверстие диаметром 4 мм. Это обеспечит плотную посадку на вал двигателя.

Катушка также имеет то преимущество, что она удерживает наматываемый трос вблизи центральной оси Y-рамы, предотвра-


**Рис. 10.13.** Лебедка Y-рамы, прикрученная к небольшой прямоугольной ПВХ-панели, монтируется посередине передней панели. На выходном валу двигателя установлена катушка. Трос от нее идет на Y-раму



щая тем самым возникновение боковых сил, стремящихся сбросить X-раму с направляющих.

Перед установкой катушки может потребоваться обрезать выходной вал двигателя для экономии места на передней панели.

Узел двигателя и коробки передач имеет по обе стороны от выходного вала два нарезных отверстия M3 (см. рис. 10.13), с помощью которых он крепится болтами к небольшой ПВХ-панели, фиксируемой на передней панели двумя скобами.

В случае использования более скоростного двигателя катушка может оказаться излишней. В таком случае трос наматывается непосредственно на вал. Для этого он фиксируется на валу путем завязывания на его конце узла с последующим пропусканием конца троса с узлом через нагретый отрезок пластиковой трубки длиной 5 мм и диаметром 3 мм. Трубка надевается на вал и после остывания плотно его охватывает. Узел не дает тросику сокочить с вала.

Если трос наматывается непосредственно на вал, то предпочтительно предварительно зафиксировать на валу два небольших диска или колеса на расстоянии около 10 мм друг от друга. Они будут выполнять роль ограничителей.

### Концевые выключатели

Рамы X и Y оснащены инфракрасными датчиками или магнитными датчиками на эффекте Холла, предназначенными для определения координат X и Y инструмента. Кроме того, пара микропереключателей, установленных на главной раме, используются в качестве концевых выключателей, распознающих нахождение рам X и Y в исходных позициях. Они позволяют точно вывести инструмент в передний правый угол главной рамы. Начиная от этого исходного положения, робот использует датчики для вывода инструмента в любую точку рабочей области. Другими словами, концевые выключатели обеспечивают стартовую точку, в которую можно возвратиться, если программа собьется в процессе отсчета координат.

Концевой выключатель для направления Y — это микропереключатель MS1, прикрепленный болтами на вершине передней правой стойки таким образом, чтобы Y-рама касалась его при достижении передней границы своего пути (рис. 10.14).

Переключатель MS2 привинчен болтами под правой направляющей главной рамы. От одного из концов X-рамы отходит скоба, форма которой обеспечивает контакт с микропереключателем главной рамы при прохождении под концом Y-рамы (рис. 10.15). Его вертикально загну-

тый конец нажимает рычажок MS2, когда X-рама достигает крайней правой точки своего пути.

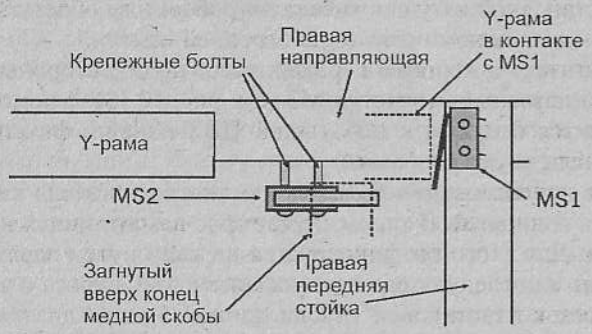



**Рис. 10.14.** MS1 и MS2 замыкаются, когда X-рама находится в переднем правом углу главной рамы

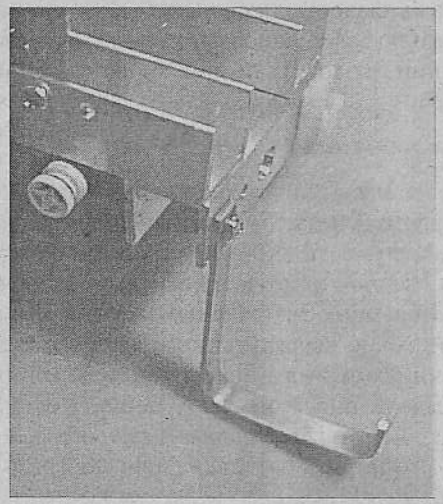


**Рис. 10.15.** Скоба, привинченная к X-раме, проходит под правой направляющей главной рамы. Ее загнутый вверх конец нажимает рычажок MS2, когда Y-рама — в переднем, а X-рама — в правом положении. Размеры скобы определяются, когда X-рама находится в исходной позиции

Таким образом, X-рама может замкнуть MS2 только тогда, когда Y-рама находится в своем исходном положении. Выполнение этого требования может взять на себя программное обеспечение. Вначале следует перемещать Y-раму до момента замыкания MS1, а затем — перемещать X-раму до момента замыкания MS2.



Обратите внимание на то, что MS1 и MS2 установлены на главной раме. Провода от них идут непосредственно на плату первого микроконтроллера PIC, управляющую приводными двигателями лебедок. Если мы не хотим, чтобы наш портальный робот превратился в хаотическое скопление проводов, идущих к рамам X и Y, необходимо свести к минимуму количество и длину соединений. Это — основная причина, по которой лебедка подъема/опускания инструмента устанавливается на Y-, а не на X-раме, где, собственно, и реализуется ее действие. По этой же причине, переключатели MS1 и MS2 расположены на главной раме на расстоянии всего лишь нескольких сантиметров от платы первого микроконтроллера PIC. Подача напряжения питания на плату второго микроконтроллера представляет собой определенную проблему. Должны ли мы размещать батарею на X-раме (занимая пространство, добавляя вес и повышая трение) или же следует проложить три провода от главной рамы (создавая риск помешать движению)? В конце концов, мы остановились на втором варианте.

### Инструменты

Тип инструмента, установленного в портальном роботе, определяется областью его применения. На X-раме можно установить только один инструмент, однако его можно легко заменить на другой, поскольку он крепится к раме болтами. Два болта M3 длиной 25 мм проходят вниз по центру длинных сегментов X-рамы.

В этом разделе рассматриваются следующие инструменты:

- **крюк** — придает портальному роботу функции подъемного крана (на поднимаемых объектах должна быть петля для зацепления);
- **схват** — относительно простая ассиметричная конструкция для подъема объектов, которая может использоваться для перемещения игровых фигур или строительства из деревянных или пластмассовых "кирпичей";
- **кисть** — используется для рисования картин;
- **лазерная указка** — используется при прохождении лабиринтов, сканировании изображений, в настольных играх;
- **камера** — замеряет яркость отраженного лазерного луча. Также применяется во множестве других сфер.

### Крюк

Крюк портального робота был выгнут из медного прута, прикрепленного к шкивному блоку (рис. 10.16). Шкивы, использованные в этом и других инструментах, были взяты из комплекта производства Tamiya.

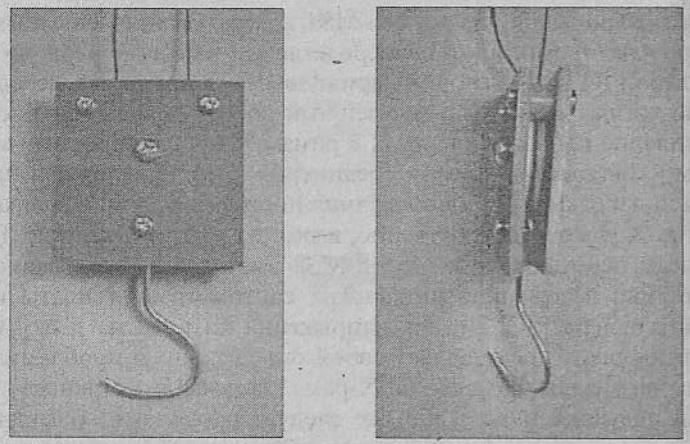


**Рис. 10.16.** Крюк, прикрепленный к шкивному блоку. Боковые панели блока скреплены у верхнего края болтами, между которыми установлены распорные втулки высотой 6 мм

Крюк поднимается и опускается специальным механизмом, который используется и для подъема и опускания схвата. Принцип его работы иллюстрирует рис. 10.17.

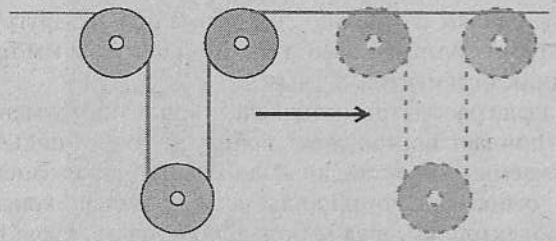


**Рис. 10.17.** Подвижный шкивный механизм. Верхние два шкива смонтированы в кожухе, прикрепленном к X-раме. Нижний шкив расположен в шкивном блоке, несущем крюк. Когда рама перемещается слева направо, высота крюка над землей не изменяется

Двигатель лебедки M3 установлен на левом краю Y-рамы. От лебедки трос проходит под X-рамой и привязывается к правому концу Y-рамы. Крюк поднимается при наматывании троса и опускается при его разматывании, однако, если рама перемещается без наматывания или разматывания троса, то крюк остается на той же высоте. Детали данного механизма и способ прохождения через него троса показаны на рис. 10.18.

Робот должен знать, несет ли коюк груз, и какая его высота над рабочей поверхностью. Эту информацию просто получить с помощью двух микропереключателей (MS1 и MS2 на рис. 10.18).



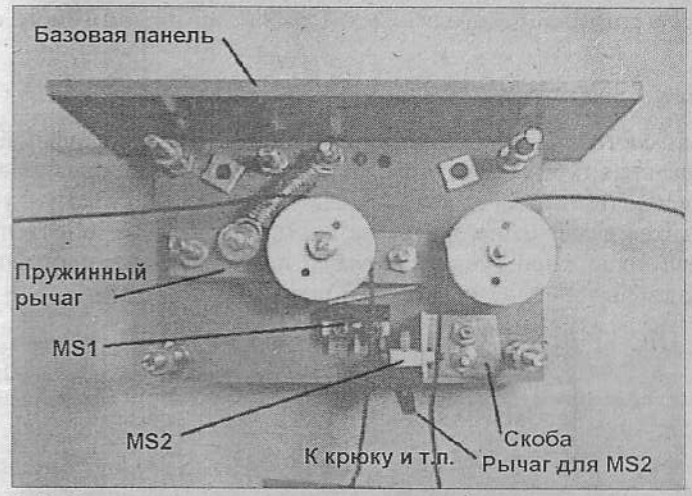



**Рис. 10.18.** Подвижный шкивный механизм с передней крышкой, снятой с четырех длинных угловых болтов. Задняя панель установлена на скобах под прямым углом к базовой панели инструмента, а значит расположена под X-рамой вертикально

Для распознавания груза на крюке служит переключатель MS1. Ось правого шкива проходит через отверстия, просверленные в передней и задней панелях. Ось левого шкива установлена на рычаге, который поворачивается вокруг одного из своих концов и удерживается пружиной. Верхний конец пружины привинчен болтом к одному из отверстий на задней панели. Натяжение пружины корректируется выбором отверстия. Оно устанавливается таким, чтобы пружина удерживала рычаг, когда нет никакой нагрузки, однако при наличии нагрузки могла растягиваться, позволяя рычагу опустится. Когда рычаг опускается, он нажимает рычажок MS1, который замыкает ключ, посылая тем самым сигнал в микроконтроллер.

Из рис. 10.18 видно, что MS2 работает как концевой выключатель. Когда шкивный блок поднят на максимально возможную высоту, он нажимает рычажок MS2, замыкая ключ, что указывает на достижение крюком его верхнего предела. Из этой позиции короткий пакет импульсов, посылаемый на приводной двигатель лебедки, выводит крюк на предсказуемую высоту.

Для обнаружения опускания крюка на уровень земли не используется никакого концевого выключателя, однако при касании груза земли переключатель MS1 будет разомкнут, а значит робот должен освободить крюк от груза, сместив X-раму в сторону. Это — пример воспол-

нения отсутствия концевого выключателя за счет программного обеспечения.

### Схват

Схват состоит из одной фиксированной и одной подвижной губки. Фиксированная губка выгнута из латунной полоски, которая затем была привинчена болтом к базовой панели. Подвижная губка — двойная и приводится в движение двигателем со встроенной коробкой передач. Два плеча подвижной губки напрямую прикреплены к противоположным концам выходного вала коробки передач (рис. 10.19).


**Рис. 10.19.** Асимметричный схват открывается и закрывается двигателем через коробку передач. Он поднимается и опускается лебедкой М3, установленной на Y-раме

Схват поднимается и опускается тем же шкивным блоком, что и крюк, с применением двух концевых выключателей.

Узел двигателя — компактный (примерные габариты 40×64×20 мм), с отверстиями для привинчивания к ПВХ-панели и зажимом для удерживания двигателя. Передаточные шестерни — пластмассовые, плотно надеваются на валы. На противоположных концах выходного вала закреплены два приводных рычага (рис. 10.20). Узел двигателя привинчен болтами к базовой ПВХ-пластине размерами 100×45 мм, которая также держит шкив и фиксированную губку схвата.



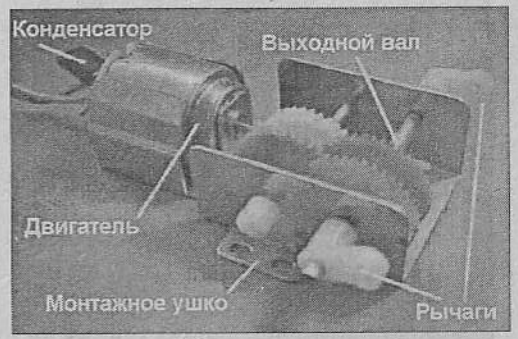



**Рис. 10.20.** Двигатель производства CIC для управления губками схвата. Подвижные губки привинчены болтами к двум рычагам, установленным на обоих концах выходного вала. Вал на несколько миллиметров выступает за рычаги для установки в два отверстия в основании подвижных губок (см. рис. 10.21)

Каждая из подвижных губок состоит из двух частей. Основание выполнено из латунной полосы шириной 5 мм и длиной 60 мм. Перед изгибанием полосы в ней были просверлены два отверстия на одном конце для крепления болтами к рычагу привода и два отверстия на другом конце для крепления зуба. Зуб выполняется из более тонкой латунной полоски 12 мм в ширину и 80 мм в длину. Фиксированная губка выполнена из латунной полосы 18 мм в ширину и 120 мм в длину. Общая длина схвата составляет 130 мм. Для некоторых операций может оказаться предпочтительнее, чтобы зубья и фиксированная губка были короче, чем показанные на рис. 10.19.

Требуемая упругость схвата обеспечивается пружинными свойствами латунных полос, используемых в фиксированной и подвижных губках. И все же необходимо обеспечить обратную связь с микроконтроллером с помощью третьего концевого выключателя. Микропереключатель устанавливается на скобе, вырезанной из узкой латунной полосы, которая позиционируется таким образом, чтобы основание губки замыкало ключ, когда зубья надежно охватывают объект (рис. 10.21). Позицию срабатывания выключателя можно отрегулировать путем сгибания скобы.

Концевой выключатель обеспечивает одну точку отсчета для определения состояния губок. Вместо установки второго концевого выключателя, повышающего сложность и вес инструмента (не говоря уже о дополнительной паре проводов), мы отталкиваемся от одной позиции, начиная с которой губки открываются до определенного уровня путем подачи на двигатель импульсов контролируемой длительности.


**Рис. 10.21.** Микропереключатель на скобе. Основание губки крепится к рычагу болтом и выходным валом, проходящим через соответствующее отверстие

Схват поддерживается под X-рамой тем же шкивным механизмом, что и в случае с крюком. Схват тяжелее крюка, поэтому точка крепления пружины должна быть изменена для увеличения натяжения.

### Кисть

Этот инструмент превращает нашего портального робота в художника. Его базовая панель привинчивается болтами прямо к днищу X-рамы. Этот инструмент содержит собственный шкивный механизм для подъема и опускания кисти (рис. 10.22) и приводится в действие лебедкой M3, установленной на Y-раме.

Кисть монтируется на четырехсторонней раме, скрепленной в углах подвижными шарнирами. Это позволяет кисти двигаться вверх и вниз, оставаясь в вертикальном положении. Когда кисть опускается вниз, выходя в позицию рисования, она слегка прижимается к бумаге, после чего перемещается и рисует линию по мере движения рам X и Y. Для прекращения рисования кисть поднимается вверх. О том, что кисть достаточно высоко поднята над бумагой, сигнализирует концевой выключатель. Теперь ее можно переместить к емкости с краской.

### Лазер

Лазер проецирует интенсивное, но очень маленькое пятно на полу под X-рамой (рис. 10.23). Оно предназначено для указания объектов на-



подобие игровых фигур. Например, лазерный луч может указывать на фигуру, которую робот собирается передвинуть. Или же он может отслеживать путь в лабиринте.

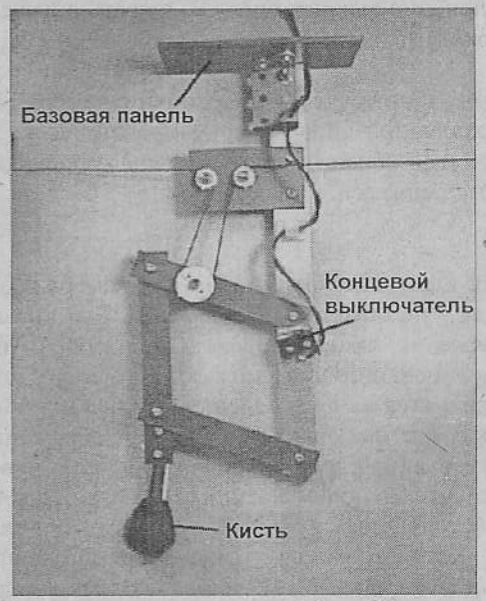



**Рис. 10.22.** Кистевой инструмент представляет собой комбинацию алюминиевой полоски, полос из ПВХ и деталей конструктора Meccano. В нем также использованы шкивы производства Tamiya и подрезанная макияжная кисть. Настоящий гибрид!

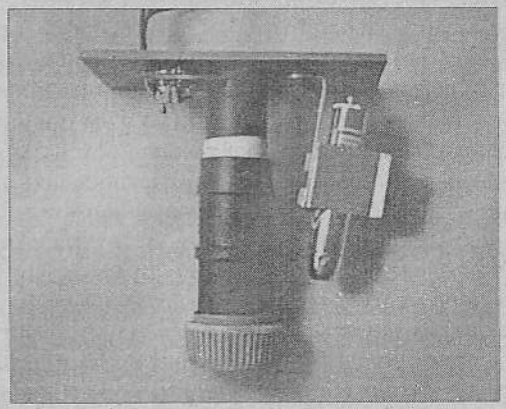


**Рис. 10.23.** Для многих приложений, связанных со сканированием, камера и лазер монтируются на одной базовой плате

В качестве лазера использована недорогая "лазерная указка", снабженная собственной батареей. Она удерживается пружинным зажимом того же типа, что и для фиксации элементов PP3 на 9 В. Указка оснащена кнопкой включения/выключения. В нашем случае указка плотно удерживается зажимом таким образом, чтобы ее кнопка включения всегда была нажата.

Этот инструмент может быть объединен с камерой (см. следующий подраздел), направленной вниз и сфокусированной на лазерном пятне. Малые размеры пятна обеспечивают высокое разрешение инструмента при обнаружении предметов.

### Камера

Используется однопиксельная цифровая камера с одним фоторезистором в качестве чувствительного элемента. Его выходной сигнал — аналоговое напряжение, изменяющееся в диапазоне от нескольких милливольт при фокусировании на черном объекте до нескольких вольт, когда камера фокусируется на ярко освещенном белом объекте.

Поскольку камера для фокусирования изображения объектов на датчике использует линзу, она обладает высокой чувствительностью и характеризуется узким полем зрения. Это важно, когда портальный робот играет в настольную игру или при регистрации схемы нарисованного лабиринта. Хотя эта камера была спроектирована специально для портального робота, она может использоваться и в других роботах.

Объектив камеры представляет собой недорогую линзу диаметром около 25 мм и фокусным расстоянием около 30 мм. Чем большее ее диаметр, тем более чувствительным будет инструмент. Линза удерживается на конце пластмассовой трубки, скользящей по неподвижной трубке (рис. 10.24). Трубки изготовлены из черной пластмассы (их можно изготовить, например, из деталей садовой поливочной системы). Стопорное кольцо также взято из крепежного узла поливочного шланга. После фокусировки подвижная трубка должна оставаться в установленной позиции, поэтому ее необходимо плотно посадить на фиксированную трубку. Предотвратить проскальзывание поможет виток или два ПВХ-изоленты вокруг конца фиксированной трубки. Аналогичным образом слой ПВХ-изоленты, намотанной вокруг фокусирующей трубки, помогает удержать на месте стопорное кольцо.

Рассмотрим оптику камеры. Линза с фокусным расстоянием *f* фокусирует полноразмерное инвертированное изображение на фоторезисторе, когда последний находится на удалении двух фокусных расстояний за линзой, а объект — на том же расстоянии перед ней. Так, например,



фокусное расстояние линзы, использованной в опытном образце, составляет 30 мм. Расстояние между фоторезистором и линзой — 60 мм. Камера монтируется таким образом, чтобы линза находилась на высоте 60 мм над рабочей областью.

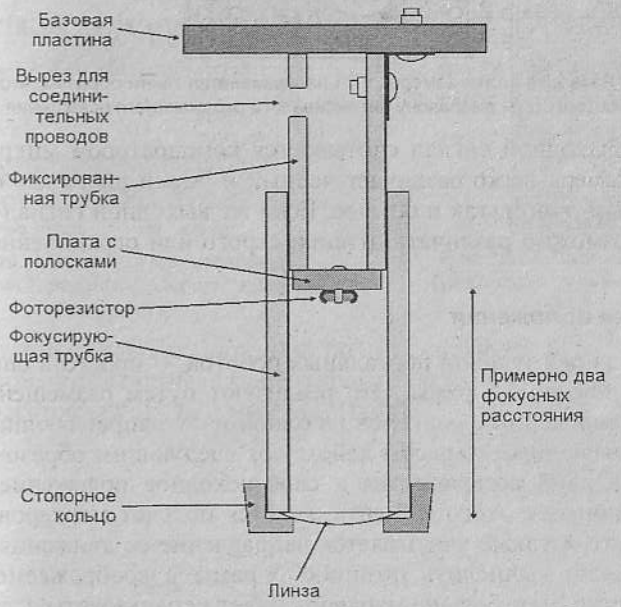



Рис. 10.24. Камера, направленная вниз для фокусирования на объекте в рабочей области портального робота

При линзе, сфокусированной на *2f*, изображение площадью 40 мм$^2$ на игровой доске полностью покрывает фоторезистор, диаметр которого составляет всего лишь 8 мм. При этом нацеливать фоторезистор в точности на центр квадрата необязательно.

Фоторезистор припаивается к крошечному прямоугольнику, вырезанному из платы с полосками. Два соединительных провода припаиваются к плате на стороне проводящих полос и выводятся наружу через отверстие, просверленное в неподвижной трубке. Толщина стенок этой трубки — 4 мм, а плата приклеивается к ее обрезанному концу.

Небольшая плата с полосками, привинченная болтами к базовой пластине, содержит переменный подстроечный резистор для настройки выходного сигнала датчика (рис. 10.25). Согласно схеме, представленной на рис. 3.20, R1 = 47 кОм, а VR1 = 470 кОм.

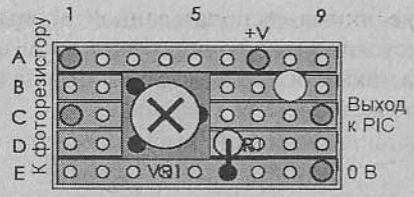



**Рис. 10.25.** Разводка платы камеры. VR1 настраивается таким образом, чтоб достичь соответствия диапазону интенсивности окружающего освещения

Когда выходной сигнал считывается компаратором микроконтроллера PIC, камера легко различает черный и белый цвета как с подсветкой лазерным лучом, так и без нее. Если же выходной сигнал подать на АЦП, то возможно различать оттенки серого или ограниченный диапазон цветов.

### Датчики положения

Одно из преимуществ портальных роботов — простота определения точного положения X-рамы. Это реализуют путем размещения равномерно распределенных маркеров на одной из X-направляющих и одной из Y-направляющих. Маркеры действуют следующим образом. В начале сеанса X-рама возвращается в свое исходное положение (спереди справа). Начиная с этого момента, ведется подсчет маркеров, которые она проходит, а также учитывается направление ее движения. По этим данным можно вычислить позицию X-рамы в воображаемой прямоугольной сетке. Чем больше маркеров будет использоваться, тем точнее будут координаты рамы.

Применяют маркеры двух типов: магнитные и инфракрасные. Мы воспользовались магнитными маркерами, однако инфракрасные были бы не менее эффективны.

Магнитные маркеры представляют собой небольшие ферриты, равномерно распределенные вдоль двух направляющих. Они распознаются датчиками Холла, смонтированными на рамах X и Y (рис. 10.26). Мы использовали квадратные магниты со стороной 10 мм и толщиной 4 мм. Приклеив их с одинаковыми интервалами к двум ПВХ-полосам, мы поместили одну такую полосу в желоб одной из X-направляющих, а другую — в желоб одной из Y-направляющих. При этом следует убедиться, что все магниты ориентированы вверх одним и тем же полюсом.

Два датчика монтируются на одном из концов Y-рамы и на одной из сторон X-рамы. Для этого каждый из них необходимо предварительно припаять к маленькому прямоугольнику, вырезанному из платы с плоскими (рис. 10.27).




**Рис. 10.26.** Маркеры для отслеживания положения X-рамы (т.е. координат X и Y инструмента)

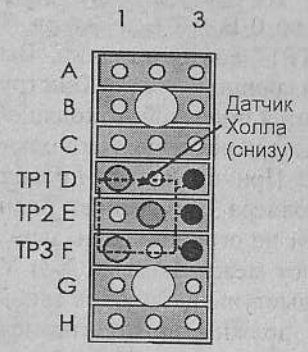



**Рис. 10.27.** Платы X-датчика (удерживается двумя болтами M2.5)

Скоба, вырезанная из латунной полосы, поддерживает X-датчик таким образом, чтобы он был ориентирован на магниты и находился над ними на расстоянии около 2 мм (рис. 10.28).

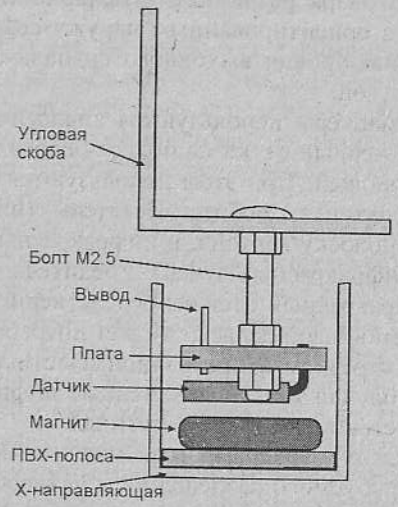



**Рис. 10.28.** Установка магнитного датчика для X-направляющей. Скоба привинчивается болтом к боковой поверхности X-рамы. Узел датчика центрируется установкой одной или двух шайб между скобой и боковой поверхностью рамы. Высота датчика устанавливается частично путем подгибания его выводов и частично — регулировкой положения платы на вертикальных болтах. Точно отрегулировать высоту можно также, поместив полоску тонкого картона под ПВХ-полосу. Провода к трем выводам проходят через отверстие в скобе и фиксируются на раме с помощью миниатюрного зажима

На плату второго микроконтроллера PIC от датчика идут три провода: 0 В, +3 В и выход. Там они подключаются к выводам TP1, TP4 и TP17 соответственно. Высота платы регулируется с помощью удерживающих ее гаек. Конструкция Y-датчика аналогична, однако удерживается полоской, отходящей от Y-рамы рядом со шкивом. Провода проходят через X-раму и подключаются к выводам TP2, TP3 и TP15.

При прокладке проводов от датчиков к плате второго микроконтроллера PIC следует планировать их маршрут таким образом, чтобы они не ограничивали движение рам. Убедитесь, что провода не провисают между рамами X и Y, поскольку они могут зацепиться за колеса и выступы наподобие скобы микропереключателя MS2. Кроме того, они не должны проходить между краем Y-рамы и микропереключателем MS1, поскольку это помешает Y-раме выйти в исходное положение.

У магнитов есть северный и южный полюса. Уровень выходного сигнала датчика повышается или падает с приближением магнита в зависимости от того, какой полюс расположен ближе. Обычно магниты приклеивают к пластмассовой полоске таким образом, чтобы вверх был обращен один и тот же полюс.

Кроме того, благодаря различиям в полярности, можно получить два типа маркеров: с ориентированным наружу северным или южным полюсом. Повышению уровня выходного сигнала соответствует один полюс, а спаду — другой.

Инфракрасные маркеры используются аналогичным образом. Они представляют собой черные метки на полосе белого картона, размещенной внутри направляющей. При этом используются датчики, аналогичные инфракрасным датчикам робота "Искатель" (инфракрасный светодиод, освещающий полоску картона, и инфракрасный фотодиод, распознающий наличие инфракрасных лучей). Убедитесь, что на фотодиод не будет попадать инфракрасный свет непосредственно от светодиода или отраженный от алюминиевой рельсы. Может потребоваться закрыть фототодиод коротким отрезком непрозрачной пластиковой трубки, чтобы он гарантировано получал только отраженные инфракрасные лучи. Такой датчик монтируется так же, как магнитный.

Промежутки между маркерами (как магнитными, так и отражающими) зависит от требуемого разрешения и размеров рабочей области. Пожалуй, наилучшие результаты будут получены с помощью сетки 8×8, как для шахматной доски (рис. 10.29). Вначале следует определить размеры рабочей области. Размер в направлении Y — это расстояние, проходимое Y-датчиком, когда рама перемещается из своего исходного положения до заднего края главной рамы. Измерим это расстояние и назо-



вем его Y. Значение X определяется аналогичным образом. Для упрощения программирования лучше всего, чтобы сетка была квадратной и покрывала рабочую область не полностью.

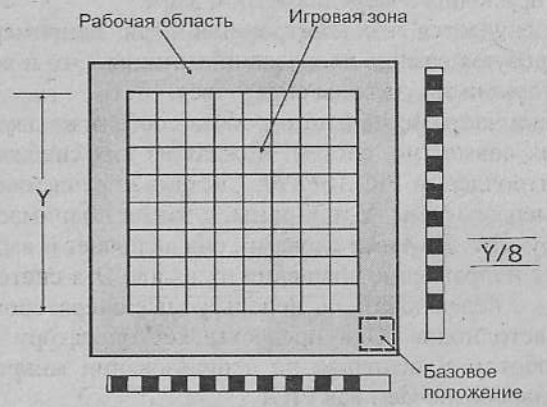



**Рис. 10.29.** Размещение маркеров. Для игровой сетки 8×8 потребуется по девять маркеров для каждого направления

В игровой программе это позволяет X-раме перемещаться за сетку (например, для того, чтобы взять или положить фигуру, которая не принимает в данный момент участия в игре в пределах сетки). В программе рисования вне области рисования можно разместить емкость с краской. Исходное положение рамы в переднем правом углу также должно находится за пределами сетки.

На рис. 10.29 видно, что количество маркеров на каждой стороне на единицу превышает число квадратов на стороне сетки. Первый маркер, начиная от исходного положения, сообщает роботу о том, что X-рама вошла в игровую область. Последующие маркеры сигнализируют о перемещении от одной строки (или столбца) к следующему. Последний (девятый) маркер сообщает роботу о том, что он вышел за пределы сетки. Для безопасности может использоваться и десятый маркер, останавливающий раму, когда она достигает края рабочей области.

Размеры X и Y должны быть меньше, чем размеры рабочей области. В случае квадратной сетки X = Y. Маркеры следует размещать с шагом X/8 и Y/8 вдоль направляющих X и Y (см. рис. 10.29).

## Электроника

В любом новаторском проекте некоторые элементы и компоненты почти неизбежно окажутся несовместимыми с другими элементами

и компонентами. Мы уже вскользь упоминали об этом в разделе "Механика". Так, может оказаться, что двигатель с диаметром выходного вала 2 мм должен приводить в действие систему передаточных шестерен, которая требует приводного вала диаметром 3 мм.

Подобное случается и с электроникой. Так, например, различные компоненты требуют разных напряжений питания. Это и есть первая из проблем, с которыми мы должны справиться.

Электронная часть портального робота состоит из двух отдельных, но работающих совместно, систем. Каждая из них снабжена собственным микроконтроллером PIC 16F690. Система управления портальным роботом перемещает рамы X и Y-рамы, а также поднимает и опускает рабочий инструмент. Другими словами, она включает и выключает двигатели и задает направление вращения их валов. Эта система также содержит кнопки и переключатели, используемые оператором, и пару индикаторных светодиодов. Она предоставляет оператору возможность управления роботом и основана на использовании микроконтроллера PIC, который мы обозначаем как PIC1.

Другая система, управляемая вторым микроконтроллером (PIC2), отвечает за работу инструмента. Именно она принимает важные решения. Эта система оснащена датчиками, на основании показаний которых она решает, что делать дальше. Она соединена с микроконтроллером PIC1, сигнализируя ему о необходимости включения двигателей. Эту двухстороннюю связь мы рассмотрим позже.

### Система управления портальным роботом

Система управления портальным роботом состоит из двух основных элементов:

- микроконтроллер PIC1 — работает от постоянного напряжения в диапазоне 2,0..5,5 В и нуждается лишь в нескольких десятках миллиампер для управления светодиодами;
- три двигателя — работают от постоянного напряжения 12 В, потребляя ток вплоть до 500 мА. При большой нагрузке они могут потребовать больше тока, однако 500 мА — наиболее вероятное значение.

Наилучший источником питания для микроконтроллера PIC — батарея из четырех никель-металл-гидридных перезаряжаемых элементов. Она дает напряжение 4,8 В (сразу же после зарядки — чуть больше). Две такие батареи удобны для обеспечения питания двигателей. В сумме они дают 9,6 В, чего немного недостаточно. В качестве альтернативы можно использовать батарею на 12 В или сетевой блок питания. Напря-



жение стабилизировать необязательно, однако блок питания должен обеспечивать ток в 500 мА.

### Плата управления питанием

Плата управления питанием монтируется на левой боковой панели портального робота, ближе к его передней части. Для подключения батарей или блока питания используется трехконтактный винтовой клеммный блок: двигатель — 9,6 В или 12 В; общая шина — 0 В; логика — 4,8 В (рис. 10.30).

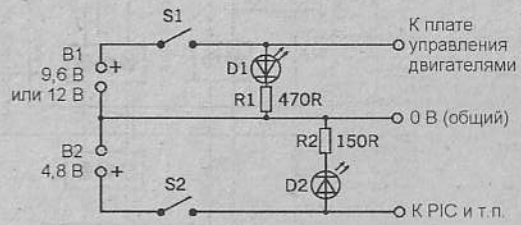



**Рис. 10.30.** Электропитание PIC1. Обратите внимание на общую линию 0 В.

Светодиоды и переключатели монтируются на боковой панели на маленькой плате, обеспечивающей подключения (рис. 10.31).

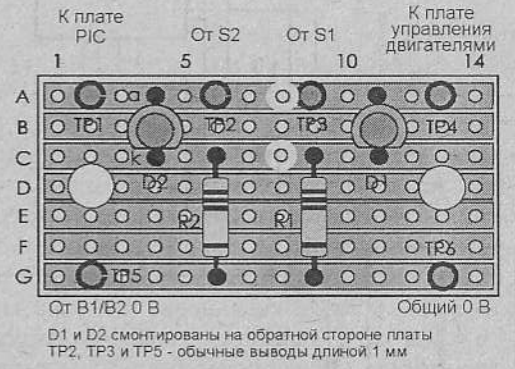



**Рис. 10.31.** Разводка платы управления питанием для системы PIC1. Эта плата крепится болтами к внутренней части левой боковой панели. Длина болтов — 25 мм, а светодиоды, размещенные в задней части платы, проходят через отверстия диаметром 5 мм, просверленные в панели

Такая организация минимизирует количество проводов, идущих от главной рамы к Y-раме. Это очень важно по той причине, что относительная жесткость проводов для такого небольшого портального робота выше по сравнению с полномасштабными промышленными портал-

ными роботами. Кроме того, чем больше проводов, тем выше вероятность, что один из них будет зацеплен инструментом, что нарушит работу робота. Полная схема линий питания показана на рис. 10.32.

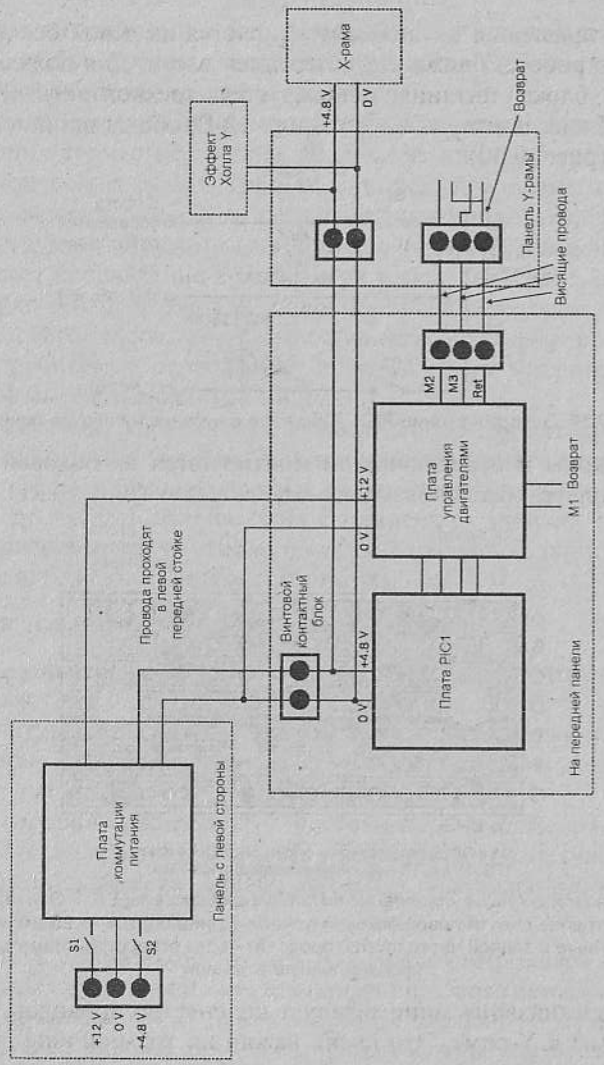



**Рис. 10.32.** Схема питания. Здесь также показаны линии управления, идущие от платы PIC1 к плате управления двигателями, а также выходные линии от этой платы к двигателям. Линия возврата — общая для всех трех двигателей



### Плата микроконтроллера PIC1

На этой плате установлен микроконтроллер PIC 16F690, который управляет приводными двигателями лебедок (рис. 10.33). Она также содержит кнопки и двухпозиционный переключатель для ввода сигнала оператором. Здесь также есть два светодиода (красный и зеленый), а также зуммер для подачи простого звукового сигнала.

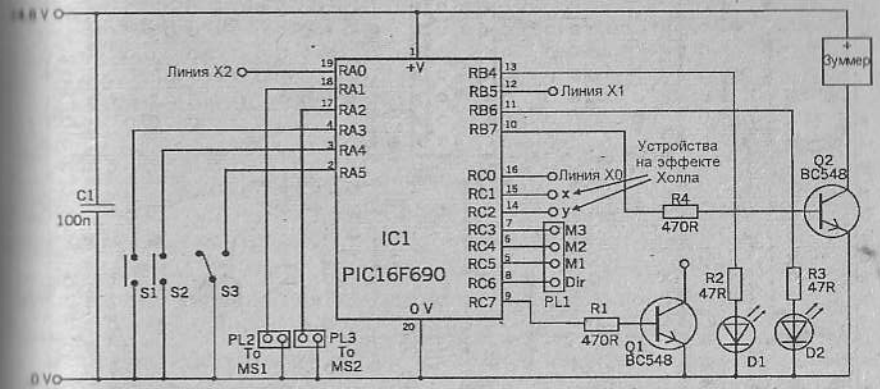



**Рис. 10.33.** Схема платы микроконтроллера PIC1

Эта плата подключается проводами к плате микроконтроллера PIC2. Для этого используются три провода: сигнал от PIC1 к PIC2, сигнал от PIC2 к PIC1 и 0 В. Эти провода идут от передней панели к X-раме. В процессе работы робота они могут натягиваться, поэтому используйте максимально гибкие провода. Они проложена от платы PIC1 вдоль внешней стороны правой направляющей к дальнему правому углу портального робота, а затем заворачиваются вокруг X-рамы.

Кроме реле управления двигателями и микропереключателей MS1 и MS2, которые подключаются к плате через разъемы PL1–PL3, все другие компоненты системы управления портальным роботом размещены на плате микроконтроллера, показанной на рис. 10.34 и рис. 10.35. Входные сигналы на микроконтроллер PIC поступают от кнопок S1 и S2, а также — от двухпозиционного переключателя S3. Выходные сигналы от PIC идут на светодиоды и зуммер (AWD). Присутствует также транзисторный ключ Q1, применяемый для включения устройств, потребляющих более тех 20 мА, которые могут быть сняты непосредственно с выхода микроконтроллера PIC.

Напряжение питания 4,8 В поступает от четырех элементов NiMH или от сетевого блока питания на 4,5 В или 6 В. Маломощный блок пи-

тания (например, на 200 мА или 300 мА) вполне приемлем, однако у не го должен быть стабилизированный выход.

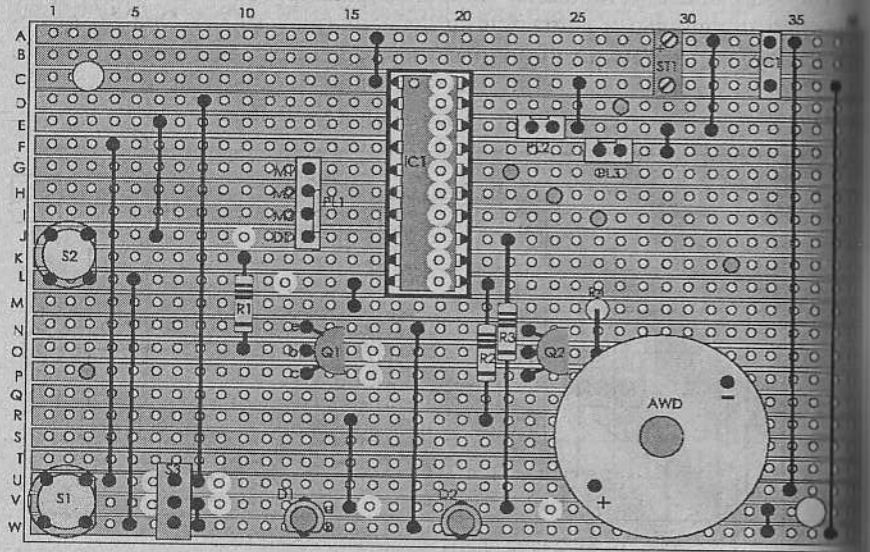



**Рис. 10.34.** Разводка платы микроконтроллера PIC1. Печатные проводники между E23/F23 и F26/F27 должны быть перерезаны

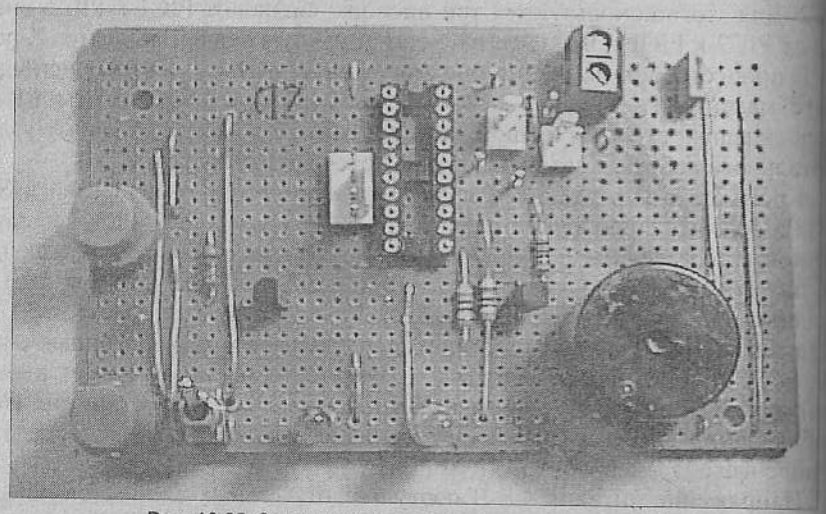


**Рис. 10.35.** Завершенная плата микроконтроллера PIC1



При сборке этой платы не должно возникать никаких проблем. Возможно, потребуется слегка модифицировать некоторые детали размещения компонентов (например, чтобы принять во внимание разницу в размещении выводов элементов, наподобие кнопочных переключателей и зуммера).

В позициях C3 и V36 присутствуют два отверстия диаметром 3 мм, предназначенные для болтового крепления платы к передней панели. В опытном образце эта плата была размещена у правого края передней панели. На болты мы установили распорки длиной 4 мм.

### Система управления инструментом

Все инструменты, а также датчики, используемые при управлении инструментами, находятся под управлением второго микроконтроллера PIC (назовем его PIC2). Его плата размещена на X-раме. Вторая функция этого микроконтроллера — отслеживание позиции X-рамы.

На раме может быть установлен только один инструмент, поэтому большинство каналов ввода-вывода микроконтроллера PIC2 связано с выводами. Выводов в наличии — более чем достаточно для обеспечения работы с инструментами, описанными выше, а также — с любыми другими, которые можно спроектировать и установить самостоятельно.

Для экономии пространства на X-раме плата включает в себя схему управления двигателями, используемую для перемещения губок схвата. Она также может быть использована для управления и другими инструментами.

Для еще большей экономии места на X-раме данная плата берет напряжение 4,8 В от батареи B2, обеспечивающей питанием систему управления портальным роботом. Это дает выигрыш с точки зрения не только пространства, но и веса. Слишком большой вес X-рамы усиливает трение в колесных узлах, что окажет влияние также и на Y-раму. Пара соответствующих проводов идет от контактного блока у левого края передней панели. Они петлей проходят через панель Y-рамы, возвращаясь к X-раме. Провод 0 В — общий для обеих систем, что существенно, если сигналы должны передаваться от одного микроконтроллера PIC к другому.

Организовать раздельное питание двигателей — задача крайне сложная. Кроме того, в этом нет необходимости, поскольку такое решение имеет смысл только при работе со схватом. Мы воспользовались схемой с общим напряжением питания с микроконтроллерм PIC, согласившись на небольшой риск возникновения пиков напряжения. Мы использовали двигатель на 3 В, который довольно хорошо переносит уве-

личение напряжение до 4,8 В, однако при желании можете воспользоваться и двигателем на 6 В, чтобы работать в режиме неполной нагруженности.

На схеме, показанной на рис. 10.36, видно наличие множества неиспользуемых выводов микроконтроллера PIC, которые можно задействовать как выходы или входы. Это обеспечивает дополнительную гибкость при проектировании и добавлении датчиков и исполнительных устройств к X-раме.

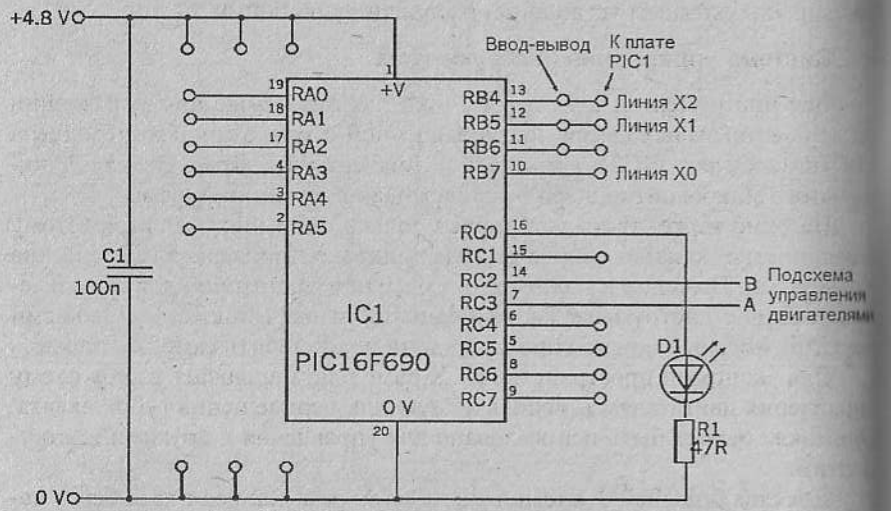



**Рис. 10.36.** Схема платы микроконтроллера PIC2

Под организацию взаимодействия с микроконтроллером PIC1 выделено три вывода. Светодиод, в основном, используется при тестировании программ. Он может временно программироваться на включение или мигание в некоторых точках программы, подтверждая тем самым, что микроконтроллер PIC2 добрался до них. Выходы магнитных датчиков X-рамы и Y-рамы соединены непосредственно с микроконтроллером PIC1.

Разводка платы микроконтроллера PIC2 показана на рис. 10.37, а в завершенном виде она представлена на рис. 10.38.

### Плата управления двигателями

Эта плата монтируется на передней панели портального робота и управляет тремя приводными двигателями лебедок. Вместо H-образного моста, она построена на четырех реле (рис. 10.39).



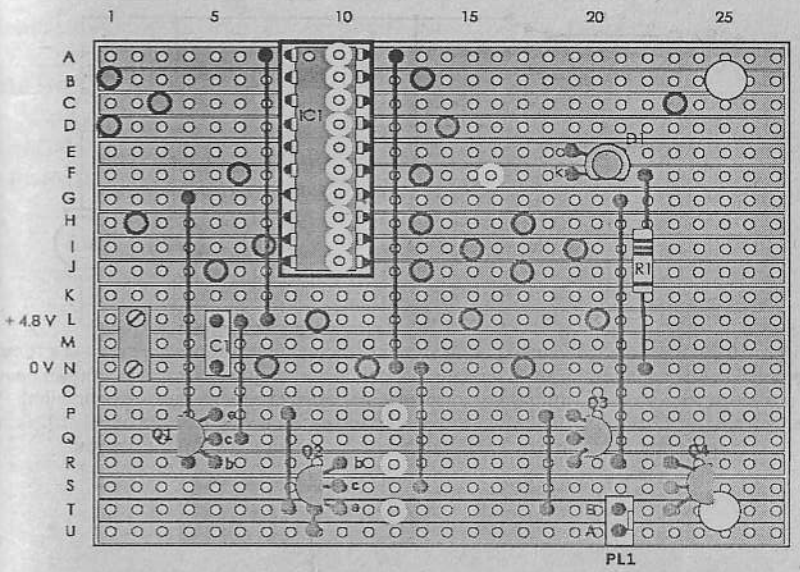



**Рис. 10.37.** Размещение элементов на плате микроконтроллера PIC2. На ней достаточно место для дальнейших расширений. Два отверстия у правого края платы предназначены для ее крепления болтами к X-раме. При этом проводящие полосы ориентированы вертикально, а пустая область справа закрывается рамой (см. рис. 10.38). Если в дальнейшем к системе предполагается добавлять другие подсхемы, то можете отрезать пустую часть данной платы и использовать ее для них

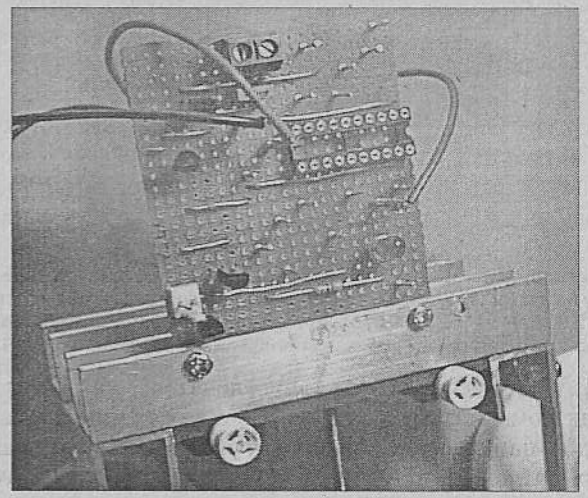


**Рис. 10.38.** Плата микроконтроллера PIC2, смонтированная на боковом сегменте X-рамы

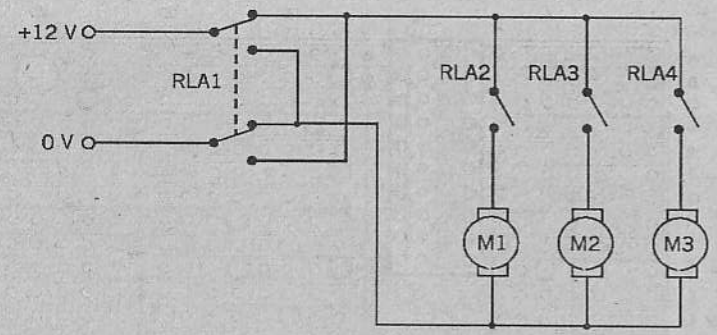



**Рис. 10.39.** Схема управления двигателями лебедок. Двигатели подключены к трем однополюсным переключателям на одно направление, однако разделяют общую линию для возврата тока в коммутатор (двухполюсный переключатель на два направления). Реле рассчитаны на 12 В (рис. 10.40), однако использованные в опытном образце устройства хорошо работали и от 9,6 В, снимаемых с четырех NiMH-элементов

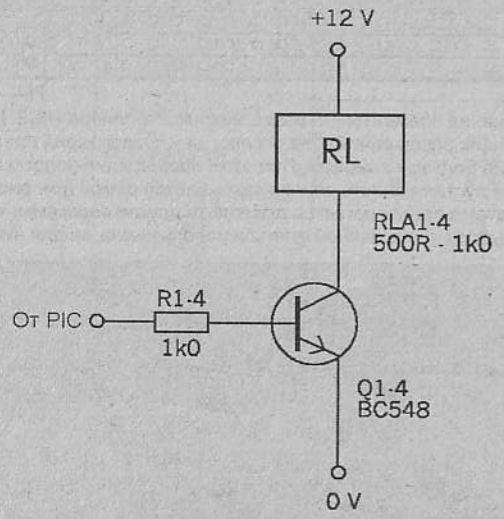



**Рис. 10.40.** Четыре реле управляются четырьмя транзисторными ключами, подобными показанному на этой схеме. Минимальное рабочее напряжение общедоступных реле составляет 12 В. Данная схема позволяет переключать реле микроконтроллером PIC, работающим при низком напряжении питания (например 4,8 В)

Реле RLA1 (двухполюсный переключатель на два направления) включено как коммутатор. Три реле RLA2–4 (однополюсные переключатели на одно направление) включают или выключают двигатели индивидуально. Эта схема принимает четыре управляющих входных сигнала от микроконтроллера PIC1 (по одному для каждого реле). Она ос-



щена двухконтактным штекерным выходом на Y-лебедку (M1), а также — трехконтактным выходом на X-лебедку (M2) и лебедку инструмента (M3).

Разводка платы (рис. 10.41) демонстрирует внутренние соединения для задействованных нами реле. Использована стандартная схема выводов реле, поэтому в этом вопросе не должно быть никаких проблем.

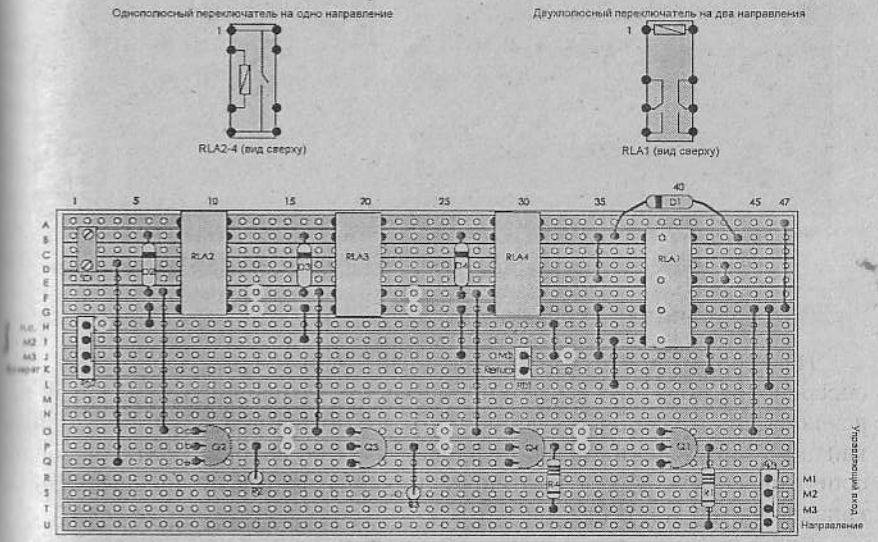



**Рис. 10.41.** Разводка платы управления двигателями. Полоса A проходит через всю плату, без разрезов. Полоса B разрезана только в точке B39. Питание 12 В или 9,6 В подается на блок винтовых клемм в точках B2 (+) и D2 (0 В)

Диоды защищают каждый транзисторный ключ от всплесков напряжения, наводимых при отключении обмоток. Подходят любые слаботочные диоды, наподобие 1N4148.

Завершенная плата управления двигателями показана на рис. 10.42.

### Схемы управления магнитными датчиками

Каждый датчик подключается к плате контроллера тремя проводами: положительная линия питания, 0 В и выход. Положительная линия питания и линия 0 В подключаются к выводам в верхней части платы микроконтроллера PIC2. Выходные провода идут непосредственно на выводы 15 (RC1) и 14 (RC2) микроконтроллера PIC1 (входы компаратора).

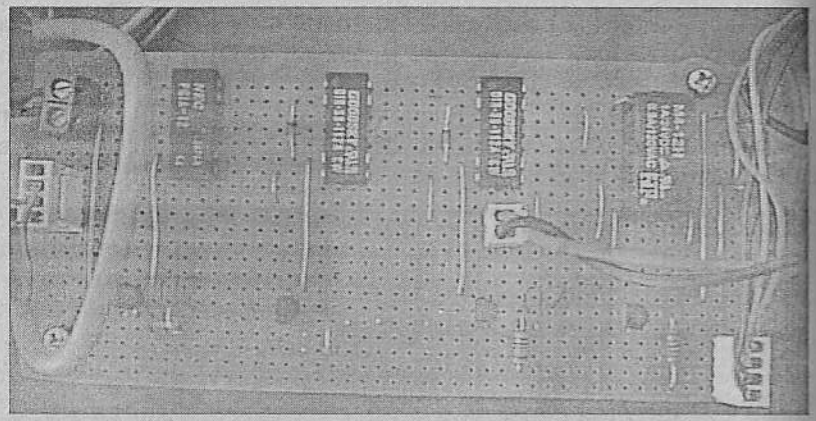


**Рис. 10.42.** Плата управления двигателями, установленная на передней панели портального робота. Справа от платы виден двигатель Y-лебедки M1

### Подключение инструментов

Некоторые инструменты оснащены микропереключателями для обеспечения обратной связи с процессором. С одной стороны такой переключатель подключен к линии 0 В, а с другой соединен со входным каналом, определенным как цифровой вход со слабым подтягивающим сопротивлением. Могут быть использованы любые незадействованные каналы портов A и B (см. табл. 10.2).

Крюк поднимается и опускается инструментальной лебедкой, взаимодействующей с двумя микропереключателями: MS1 и MS2 (см. рис. 10.18). Ключи срабатывают путем заземления входа. От шкивного блока на плату процессора идут три провода, которые заканчиваются штекерами, надеваемыми на контактные выводы печатной платы. Каждый ключ соединен проводом с микроконтроллером PIC. Третьему проводу соответствует общая "земля". Используйте один из выводов 0 В на полосе C.

Схват оснащен одним микропереключателем, который подсоединяется так же, как и микропереключатели шкивного блока: с выводом 0 В и неиспользуемым входом порта A или B (программируется как цифровой вход со слабым подтягивающим сопротивлением). Двойной провод от двигателя управления губкой M4 заканчивается двухконтактным разъемом, который надевается на выводы печатной платы процессора в позициях V24/W24.

Кисть оснащена одним микропереключателем, подключенным к линии 0 В и цифровому входу со слабым подтягивающим сопротивлени-



ем. Если к инструментам добавить другие микропереключатели, то их можно подключить к незадействованным входам портов A и B.

Лазер оснащен собственной встроенной батареей. На его плате может также устанавливаться камера, фоторезистор которой работает как делитель напряжения, в силу чего требует трех проводов: положительная линия питания, 0 В и выход. Для первых двух подключений используйте выводы на полосах A и C. Выход идет на АЦП или компаратор. Может использоваться любой из входных каналов АЦП (от AN0 до AN11) или входных каналов компаратора (от AN1 или AN5 до AN7) (см. табл. 10.2).

### Помехи

Портальный робот содержит несколько соединительных проводов, длина которых равна или превышает 10 см. Эти провода подвержены воздействию электромагнитных помех от всплесков напряжения и сигналов, излучаемых от других соединительных проводов. Так, например, длина проводов от датчиков Холла, идущих к плате микроконтроллера PIC1, составляет около 70 см. Воздействие на них помех может привести к неправильному подсчету магнитных маркеров. Такая ошибка возникает не всегда, однако при ее проявлении рамы X и Y не будут выводиться в требуемые позиции. К тому же, подобные ошибки накапливаются.

Приемлемое решение данной проблемы — воспользоваться экранированным кабелем. К сожалению, это снизит эластичность соединений. Мы обнаружили, что влияние помех устраняется, если припаять конденсатор с полиэфирным диэлектриком на 100 нФ между входными выводами платы контроллера PIC1 и линиями 0 В. На рис. 10.34 эти конденсаторы находятся в позициях C32/G32 и C38/H38.

Помехи могут также ослабить действие слабых подтягивающих сопротивлений цифровых входов, которые переходят в состояние низкого уровня в результате заземления на линию 0 В. Пример — входные сигналы концевых выключателей, которые обнаруживают момент выхода X-рамы в исходное положение. Другой пример — входы от микропереключателей инструментов. В таких случаях входы должны конфигурироваться без использования слабых подтягивающих сопротивлений. Вместо этого к ним должен подключаться внешний подтягивающий резистор, соединенный с положительной линией питания. Обычно приемлем резистор с номиналом 10 кОм, однако могут использоваться и меньшие номиналы вплоть до 1 кОм.

Соответствующая цепь показана на средней схеме на рис. 3.15. Резистор может быть размещен на инструменте или на плате контроллера.

### Выводы микроконтроллера PIC1

В табл. 10.1 перечислены подключения к выводам микроконтроллера PIC1. Ксерокопия этой таблицы, прикрепленная к стенду, будет полезной при сборке и тестировании модулей системы.

**Таблица 10.1.** Выводы микроконтроллера PIC1 в портальном роботе

| Порт | Вывод | Номер | Тип | Подключение | 0 = | 1 = |
|---|---|---|---|---|---|---|
| A | RA0 | 19 | Вход/выход | X2 | | |
| | RA1 | 18 | Вход | Микропереключатель 1 | Замкнут | Разомкнут |
| | RA2 | 17 | Вход | Микропереключатель 2 | Замкнут | Разомкнут |
| | RA3 | 4 | Вход | Кнопка S1 | Нажата | Отпущена |
| | RA4 | 3 | Вход | Кнопка S2 | Нажата | Отпущена |
| | RA5 | 2 | Вход | Переключатель S3 | Замкнут | Разомкнут |
| B | RB4 | 13 | Выход | D2 (зеленый) | Выкл. | Вкл. |
| | RB5* | 12 | Вход/выход | X1 | | |
| | RB6 | 11 | Выход | D1 (красный) | Выкл. | Вкл. |
| | RB7* | 10 | Выход | Зуммер | Выкл. | Вкл. |
| C | RC0* | 16 | Вход/выход | X0 | | |
| | RC1 | 15 | Аналог. вход | Датчик Холла (x) | Между | Над маркером |
| | RC2 | 14 | Аналог. вход | Датчик Холла (y) | Между | Над маркером |
| | RC3 | 7 | Выход | Двигатель 3 (лебедка инструмента) | Выкл. | Вкл. |
| | RC4 | 6 | Выход | Двигатель 2 (X-лебедка) | Выкл. | Вкл. |
| | RC5 | 5 | Выход | Двигатель 1 (Y-лебедка) | Выкл. | Вкл. |
| | RC6 | 8 | Выход | Двигатели (направление) | Уменьшение | Увеличение |
| | RC7 | 9 | Выход | Резервный транзисторный ключ | Выкл. | Вкл. |



Строки табл. 10.1 сгруппированы по портам ввода-вывода. В ней перечислены все каналы, присутствующие в PIC16F690. Кнопки S1 и S2 используются в играх и т.п., вводя команды типа "Старт", "Ход сделан" и т.п. Переключатель S3 обеспечивает выбор программы или выполняет другие подобные функции.

X0, X1 и X2 — это линии, идущие к плате на X-раме. Они управляют инструментом или обеспечивают обмен сигналами с микроконтроллером PIC2 при его наличии. Линии, помеченные как *, задействованы для некоторых из программ.

### Выводы микроконтроллера PIC2

В табл. 10.2 перечислены подключения к выводам микроконтроллера PIC2. Ксерокопия этой таблицы, прикрепленная к стенду, будет полезной при сборке и тестировании модулей системы.

**Таблица 10.2.** Выводы микроконтроллера PIC2 в портальном роботе

| Порт | Вывод | Номер | Тип | Подключение | Вход компаратора | Вход АЦП |
|---|---|---|---|---|---|---|
| A | RA0 | 19 | Вход/выход | Резерв | Опорное напряжение | AN0 |
| | RA1 | 18 | Вход/выход | Зарезервирован для камеры | Вход 1 или 2 | AN1 |
| | RA2 | 17 | Вход/выход | Резерв | | AN2 |
| | RA3 | 4 | Вход | Резерв (только ввод) | | |
| | RA4 | 3 | Вход/выход | Резерв | | AN3 |
| | RA5 | 2 | Вход/выход | Резерв | | |
| B | RB4 | 13 | Вход/выход | X2 | | AN10 |
| | RB5 | 12 | Вход | X1 (или вход USART) | | |
| | RB6 | 11 | Вход/выход | Резерв | | |
| | RB7 | 10 | Выход | X0 (или вход USART) | | |
| C | RC0 | 16 | Выход | Светодиод D1 (красный) | | |
| | RC1 | 15 | Вход/выход | Резерв | Вход 1 или 2 | AN5 |
| | RC2 | 14 | Выход | Двигатель A | | AN6 |
| | RC3 | 7 | Выход | Двигатель B | | AN7 |
| | RC4 | 6 | Вход/выход | Резерв | | |
| | RC5 | 5 | Вход/выход | Резерв | | |
| | RC6 | 8 | Вход/выход | Резерв | | AN8 |
| | RC7 | 9 | Вход/выход | Резерв | | AN9 |

Строки табл. 10.2 сгруппированы по портам ввода-вывода. В ней перечислены все каналы, присутствующие в PIC16F690. Каналы, которые могут быть сконфигурированы как входы компараторов и АЦП, перечислены в шестой и седьмой колонках.

X0, X1 и X2 — это линии, идущие к плате процессора PIC1 на передней панели. Они передают информацию от датчиков или принимают сигналы от микроконтроллера PIC1.

Когда каналы портов A и B работают как входы, для них могут программироваться функции прерываний по изменению состояния и слабые подтягивающие сопротивления.

*Список приобретений для электронной части робота*

Плата управления питанием:
- R1 — резистор на 470 Ом;
- R2 — резистор на 150 Ом;
- D1, D2 — светодиоды диаметром 5 мм;
- S1, S2 — миниатюрные двухпозиционные переключатели, однополюсные на одно или два направления;
- выводы 0,9 мм (6 шт.);
- плата с полосками: 7 полос x 14 отверстий.

Плата микроконтроллера PIC1:
- R1, R4 — резисторы на 470 Ом (2 шт.);
- R2, R3 — резисторы на 47 Ом (2 шт.);
- C1 — конденсатор с полиэфирным диэлектриком, 100 нФ;
- D1, D2 — светодиоды диаметром 5 мм (красный и зеленый);
- Q1, Q2 — n-p-n-транзисторы, типа BC548 или подобные;
- AWD — полупроводниковый зуммер или сирена;
- PL1 — четырехконтактный разъем и штекер;
- PL2, PL3 — двухконтактные разъемы и штекеры (2 шт.);
- S1, S2 — кнопки для монтажа на печатной плате (2 шт.);
- S3 — миниатюрный двухпозиционный переключатель, однополюсный на одно или два направления;
- винтовые клеммы для монтажа на печатной плате;
- выводы 0,9 мм (6 шт.);
- гнездо для микросхемы, 20-контактная с двухрядным размещением выводов;
- плата с полосками: 23 полосы x 38 отверстий.

Плата управления микроконтроллером PIC2 и двигателем инструмента:
- R1 — резистор на 47 Ом;
- C1 — конденсатор с полиэфирным диэлектриком, 100 нФ;



- D1 — светодиод диаметром 5 мм;
- Q1, Q3 — n-p-n-транзисторы типа BC639 или подобные;
- Q2, Q4 — p-n-p-транзисторы типа BC640 или подобные;
- PL1 — двухконтактный разъем и штекер;
- винтовые клеммы для монтажа на печатной плате;
- выводы 0,9 мм (22 шт.);
- гнездо для микросхемы, 20-контактное с двухрядным размещением выводов;
- плата с полосками: 21 полоса x 27 отверстий.

Плата управления двигателями (для лебедок):
- R1–R4 — резисторы на 1 кОм (4 шт.);
- Q1–Q4 — n-p-n-транзисторы типа BC548 или подобные (4 шт.);
- RLA1 — двухполюсный переключатель на два направления, 12 В, миниатюрный, для монтажа на печатной плате;
- RLA2–RLA4 — однополюсные переключатели на одно или два направления, 12 В, миниатюрные, для монтажа на печатной плате (3 шт.);
- разъемы для микросхемы с двухрядным размещением выводов (14-контактные — 3 шт.; 16-контактные — 1 шт.);
- винтовые клеммы для монтажа на печатной плате;
- PL1 — двухконтактный разъем и штекер;
- PL2, PL3 — четырехконтактные разъемы и штекеры (2 шт.);
- плата с полосками: 21 полоса x 47 отверстий.

Плата камеры:
- R1 — резистор на 47 кОм;
- VR1 — миниатюрный подстроечный потенциометр, 470 кОм;
- выводы 0,9 мм (5 шт.);
- плата с полосками: 5 полос x 9 отверстий.

Внеплатные компоненты:
- LDR1 — фоторезистор (ORP12 или подобный);
- UGN3503U — магнитные датчики Холла (2 шт.);
- лазерная указка;
- ферриты 10x10x4 мм (12 шт. или больше);
- батарейный отсек 8xAAA или 8xAA с проводными или штифтовыми выводами;
- батарейный отсек 4xAAA или 4xAA с проводными или штифтовыми выводами;
- соединительный разъем батареи PP3 (для штифтовых выводов) (2 шт.);
- перезаряжаемые NiMH-элементы (12 шт.) (рекомендуется);

миниатюрные микропереключатели (2 шт. — для главной рамы; 2 шт. — для крюка; 1 шт. — для схвата; 1 шт. — для кисти).

Разное:

монтажные провода;

припой.

## Программирование

Электроника портального робота позволяет нам программировать его тремя различными способами.

- Управление от одного микроконтроллера PIC, размещенного на лицевой панели главной рамы. Двигатели M1 к M3 контролируются непосредственно платой управления двигателями. Датчики и инструменты размещены на X-раме с проводными подключениями к плате микроконтроллера PIC. Гнездо PIC на X-раме не используется.
- Управление от одного микроконтроллера PIC, размещенного на X-раме. Вариант, обратный предыдущему, и, пожалуй, — не особо практичный.
- Управление от двух микроконтроллеров PIC, работающих совместно. PIC1 главным образом занят управлением двигателями и интерфейсом с пользователем, а PIC2 управляет работой датчиков и инструментов. Для координирования работы двух микроконтроллеров используются проводные соединения.

Если на X-раме установлен ряд датчиков и исполнительных устройств, то лучше всего, если данные от них будут обрабатываться микроконтроллером PIC, расположенным на X-раме. Из трех перечисленных выше вариантов оптимальный — третий. Он не только обеспечивает прирост вычислительной мощности, но и снижает количество соединительных проводов между X-рамой и главной рамой. Это позволяет X-раме двигаться более свободно. Кроме того, снижается риск спутывания проводов. Программирование для такого случая рассматривается в главе 5.

На практике, простая система с небольшим числом датчиков может хорошо работать при ограниченном количестве проводов между рамами. Такому подходу соответствует первый из перечисленных выше вариантов. В примерах, рассмотренных в этом разделе, используются шесть линий: две питания и четыре сигнальных. Сигнальные линии предназначены для решения различных задач в зависимости от используемого инструмента и вида активности.



Программы для портального робота содержат те же подготовительные операции, что и программы для других проектов. Необходимо загрузить заголовочный файл, содержащий необходимые директивы, а затем добавить объявления меток. Эти директивы и настройки процессора показаны в листинге 10.1.

**Листинг 10.1. Директивы и настройки процессора для портального робота**

```
; метки
delay0      equ 20h
delay1      equ 21h
delayn      equ 22h
xpos        equ 23h
ypos        equ 24h
flags       equ 25h

  goto start
  org 04h
  goto start

start
  bcf intcon, 7      ; Запрет прерываний
  bcf status, 5      ; Банк0
  bcf status, 6
  clrf porta
  clrf portb
  clrf portc
  bsf status, 5      ; Банк1
  movlw 00h          ; Все каналы порта B - выходы
  movwf trisb
  movlw 06h          ; Все каналы порта C - выходы
  movwf trisc        ;  за исключением RC1 и RC2
  bcf option_reg, 7 ; Активизация слабых подтягивающих
  bcf wpua, 7        ;   сопротивлений, кроме RA0
  bcf status, 5      ; Банк2
  bsf status, 6
  movlw 60h          ; Аналоговый вход RC1,2
  movwf ansel
  clrf anselh
  movlw 95h          ; Настройка компаратора
  movwf cm1con0
  movlw 96h
  movwf cm2con0
  movlw 0c9h         ; Опорное напряжение = 0,53 x напр. питания
  movwf vrcon
  bcf status, 6      ; Банк0
  bcf status, 5
```

Каналы порта A — все входы, и поскольку они устанавливаются таким образом по умолчанию после включения питания, в отношении конфигурирования порта A нет необходимости что-либо предпринимать.

После перехода к банку 1 все каналы портов B и C — выходы, кроме RC1 и RC2, которые должны быть аналоговыми входами для сигналов от двух магнитных датчиков Холла.

Все каналы порта A требуют слабых подтягивающих сопротивлений, поскольку они подключены к переключателям или кнопкам, замыкающим канал при замыкании. После перехода к банку 2 два входных канала порта C (RC1 и RC2) объявляются как аналоговые входы.

Следующие пять строк настраивают компараторы (см. главу 4). В регистр CM1CON0 записывается значение 95h: разряд <7> активизирует компаратор 1, разряд <4> выбирает неинвертированную полярность выхода, а разряд <2> определяет использование внутреннего опорного напряжения. Разряды <1:0> устанавливаются в 01 для объявления канала RC1 (вывод 15) входом. Запись в регистр CM2CON0 значения 96h обеспечивает для компаратора 2 те же настройки, что и для компаратора 1, за исключением того, что входом является вывод RC2 (вывод 14).

Перед установкой переменного опорного напряжения, микроконтроллер PIC был изъят из его гнезда, и был включен источник питания 4,8 В. После подключения вольтметра к выходу датчика Холла 1 и линии 0 В X-рама перемещалась вручную по направляющим. При этом снимались отсчеты с датчика между магнитными маркерами и в те моменты, когда датчик находился непосредственно над маркером. При напряжении питания 5,1 В (свежезаряженные элементы), на выходе считывалось 2,5 В между маркерами с пиковым значением около 3 В над маркерами. Порог чувствительности составляет 2,75 В, что примерно равно напряжению питания, умноженному на коэффициент 0,53. Можно предположить, что по мере уменьшения напряжения питания ввиду разрядки элементов выходное и пороговое напряжения будут снижаться пропорционально.

Несколько попыток расчета на калькуляторе показали, что оптимальная настройка опорного напряжения — выбор верхнего диапазона (см. главу 4) и использование формулы $V_{пит} \times (0{,}25 + Значение/32)$. При значении 9 в разрядах <3:0> результат будет равен 0,53. Разряды <7:5> для такого опорного напряжения для обоих компараторов содержат единицы, что соответствует выбору верхнего диапазона. Таким образом, регистр VRCON содержит код c9h.



То же самое опорное напряжение должно использоваться для обоих компараторов, но это — не проблема, поскольку датчики идентичны и расположены на одинаковом расстоянии над магнитами. Тем не менее, перед принятием решения об уровне опорного напряжения рекомендуем замерить выходные сигналы с обоих датчиков.

Перед продолжением программы не забудьте вернуться к банку 0.

### Ориентация в пространстве

Первое, что необходимо знать микроконтроллеру, — местоположение X-рамы, которая может оказаться в любой позиции после предыдущего сеанса работы. Магнитные маркеры — это просто метки, наподобие километровых столбов. На них не указано расстояние. Вот почему в переднем правом углу главной рамы находятся два микропереключателя. Когда портальный робот включается, программа перемещает раму до тех пор, пока не сработают оба выключателя. После этого у микроконтроллера PIC будет надежная точка отсчета: **базовая позиция**.

Как только робот получил точку отсчета, он может перемещать X-раму по рабочей области, подсчитывая маркеры в направлениях X и Y. Теперь он всегда будет знать, где находится X-рама. В листинге 10.2 показан код для вывода X-рамы в базовую позицию (рис. 10.43).

**Листинг 10.2. Код вывода X-рамы в базовую позицию**

```
; Программа начинается здесь
gofront
  btfss porta, 1     ; MS1 замкнут?
  goto atfront       ; Да
  bsf portc, 5       ; Наматывание Y-лебедки
  bcf portc, 6
  goto gofront       ; Повторная проверка состояния MS1
atfront
  bcf portc, 5       ; Останов Y-лебедки
goright
  btfss porta, 2     ; MS2 замкнут?
  goto atbase        ; Да
  bsf portc, 4       ; Разматывание X-лебедки
  bcf portc, 6
  goto goright       ; Повторная проверка состояния MS2
atbase
  bcf portc, 4       ; Останов X-лебедки
  bsf portb, 4       ; Включение зеленого светодиода
  movlw 05h
  call longdelay
  bcf portb, 4       ; Лог. 0 для выключения зеленого светодиода
```

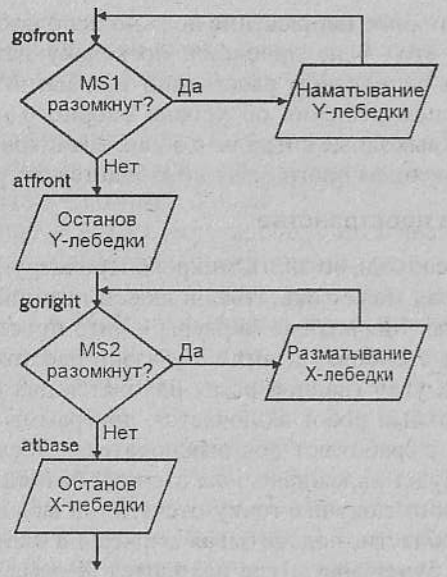



**Рис. 10.43.** Вначале перемещаем Y-раму в направлении передней панели до замыкания MS1. Затем перемещаем X-раму вправо до замыкания MS2. Наконец (необязательно) включаем зеленый светодиод для индикации базовой позиции X-рамы

Команды на лебедки X и Y подаются в виде уровней напряжения на RB4 (X-лебедка), RB5 (Y-лебедка) и RB6 (направление) (табл. 10.3).

**Таблица 10.3.** Команды для лебедок X и Y

| Направление | RB4 (X) | RB5 (Y) | RB6 (направление) |
|---|---|---|---|
| Влево | 1 | X | 1 |
| Вправо | 1 | X | 0 |
| Назад | X | 1 | 1 |
| Вперед | X | 1 | 0 |
| Стоп | 0 | 0 | X |

Для каждого направления следует установить высокий уровень на выводе RB4 (X-лебедка, M2) для движения влево/вправо или на выводе RB5 (Y-лебедка, M1) для движения назад/вперед. Направление определяет состояние вывода RB6. Для остановки X-рамы следует установить низкий уровень на выводе RB4 или RB5.

После вывода X-рамы в базовую позицию можно приступать к ее перемещению в рабочую область для решения поставленных задач. Эти задачи используют набор подпрограмм для навигации X-рамы в рабочей области.



### Перемещение X-рамы

Одно из преимуществ портального робота заключается в том, что процессор всегда знает координаты X-рамы. В силу этого одной из важных задач программирования является перемещение X-рамы в необходимую точку. Как уже объяснялось ранее, портальный робот позиционирует X-раму в квадратной сетке. В опытном образце исопльзовалась сетка 6×6, что в сумме дает 36 ячеек.

Для перемещения X-рамы используется четыре основных подпрограммы: по одной для каждого направления. В листинге 10.3 показана подпрограмма для перемещения на один шаг влево. Соответствующая блок-схема дана на рис. 10.44.

**Листинг 10.3. Подпрограмма для перемещения X-рамы на один шаг влево**

```
left
  bsf portb, 6        ; Включаем красный светодиод
  movlw 03h
  call longdelay
  bcf portb, 6        ; Выключаем красный светодиод
  bsf portc, 4        ; Наматывание X-лебедки
  bsf portc, 6
  movlw 07h           ; Задержка на 1,4 с
  call longdelay      ; Для подтверждения маркера
  bsf status, 6       ; Страница 2
againl
  btfss cm1con0, 6    ; Разряд устанавливается, если напряжение
                      ;  на входе > 0,53 х напряжение питания
  goto againl
  bcf status, 6       ; Назад к странице 0
  btfss flags, 0
  bcf portc, 4        ; Останов
  return
```

Перед вызовом этой подпрограммы следует установить или обнулить разряд <0> регистра флагов. Восемь разрядов этого регистра используются для индикации каких-либо событий или уведомления о том, что нечто должно произойти в будущем. В этой программе разряд <0> обнуляется. Это означает, что X-рама после выполнения одного шага должна остановиться. Если рама должна продолжать свое движение, то этот разряд устанавливается в 1.

Работа подпрограммы начинается с мигания красного светодиода. Это несущественно и может быть опущено, хотя и удобно при тестировании программы. Далее включается двигатель M2 для намотки троса на лебедку и перемещения рамы влево. На этой стадии отрабатывается

длинная задержка, поскольку предполагается, что X-рама остановлена на маркере. Для поиска следующего маркера она должна отодвинуться от текущего. В зависимости от скорости вращения вала и передаточного коэффициента коробки передач может потребоваться изменить длительность этой задержки. Единственный практический способ определить ее — метод проб и ошибок.

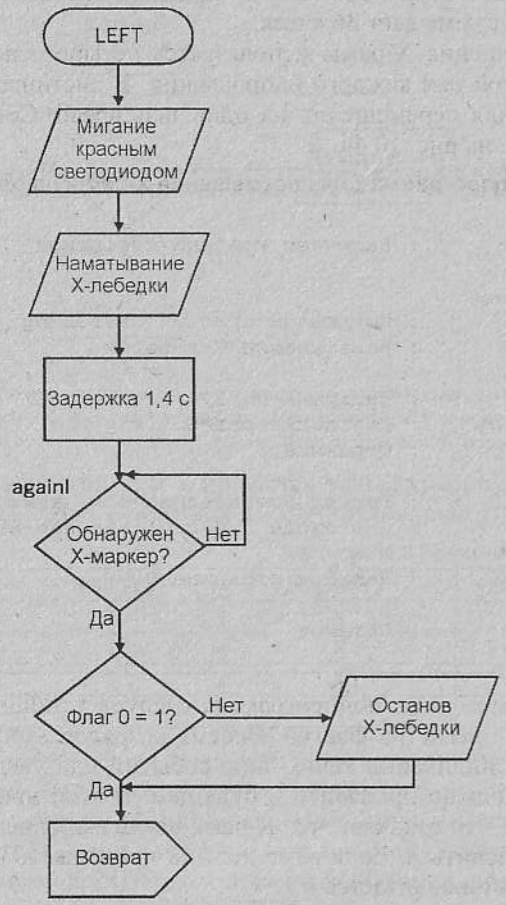



**Рис. 10.44.** Блок-схема подпрограммы `left`

Далее следует цикл, состоящий из двух командных строк, в котором разряд <6> регистра CM1CON0 опрашивается до тех пор, пока не окажется состоянии лог. 1. Обратите внимание на переключение к банку 2



перед опросом CM1CON0, а также — на обратное переключение в банку 0 после обнаружения маркера.

Подпрограмма завершается остановом лебедки или предоставлением ей возможности работать дальше в зависимости от состояния flags<0>. X-рама переместилась от одного маркера к следующему в том же ряду.

Подпрограмма `right` (листинг 10.4 и рис. 10.45) работает аналогичным образом, однако двигает раму в противоположном направлении

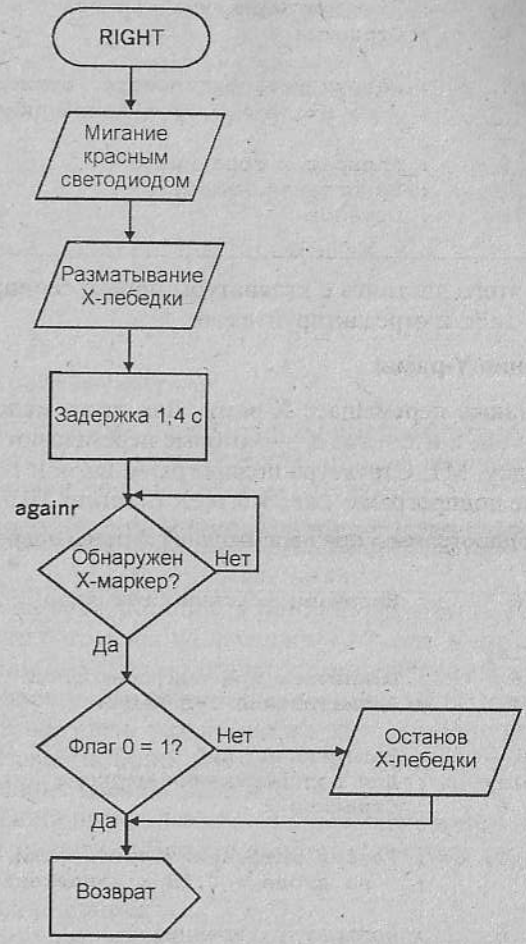



**Рис. 10.45.** Блок-схема подпрограммы `right`

длинная задержка, поскольку предполагается, что X-рама остановлена на маркере. Для поиска следующего маркера она должна отодвинуться от текущего. В зависимости от скорости вращения вала и передаточного коэффициента коробки передач может потребоваться изменить длительность этой задержки. Единственный практический способ определить ее — метод проб и ошибок.

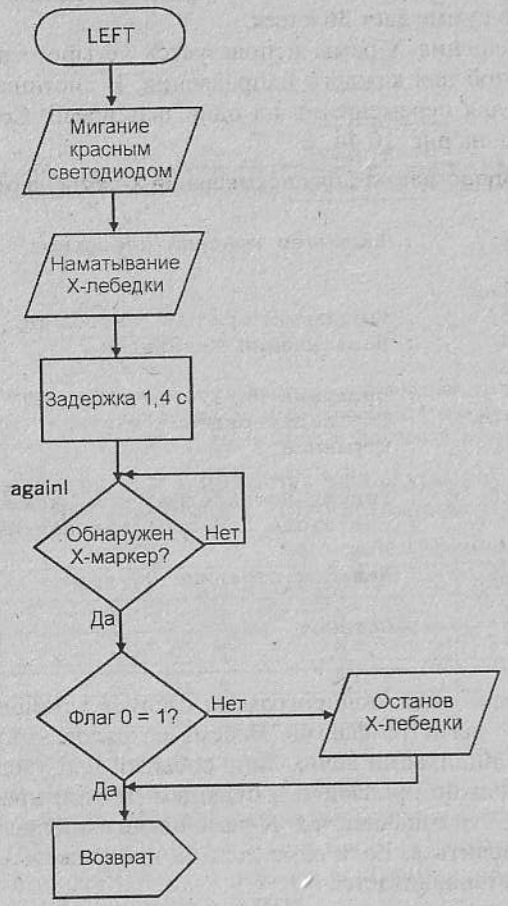



**Рис. 10.44.** Блок-схема подпрограммы `left`

Далее следует цикл, состоящий из двух командных строк, в которо разряд <6> регистра CM1CON0 опрашивается до тех пор, пока не ок жется состоянии лог. 1. Обратите внимание на переключение к банку



перед опросом CM1CON0, а также — на обратное переключение к банку 0 после обнаружения маркера.

Подпрограмма завершается остановом лебедки или предоставлением ей возможности работать дальше в зависимости от состояния flags<0>. X-рама переместилась от одного маркера к следующему в том же ряду.

Подпрограмма `right` (листинг 10.4 и рис. 10.45) работает аналогичным образом, однако двигает раму в противоположном направлении.

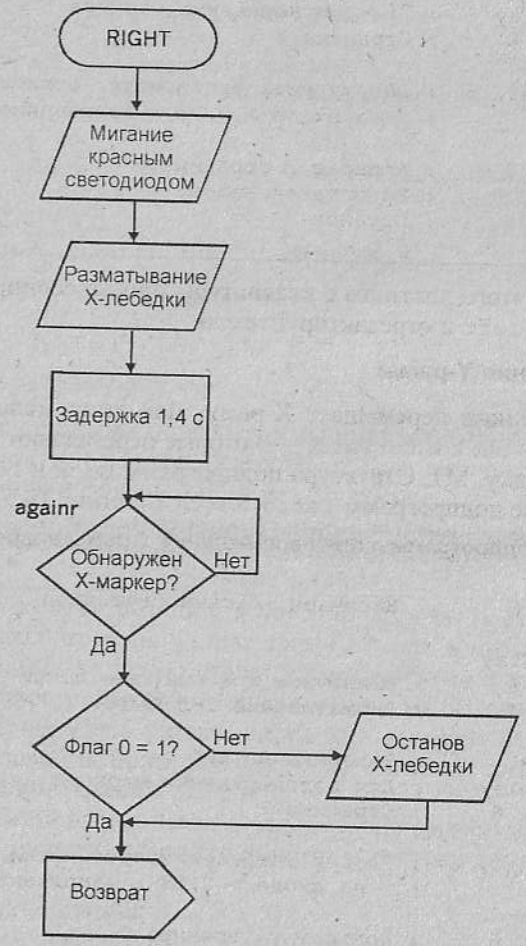



**Рис. 10.45.** Блок-схема подпрограммы `right`

**Листинг 10.4. Подпрограмма для перемещения X-рамы на один шаг вправо**

```
right
  bsf portb, 6      ; Включаем красный светодиод
  movlw 03h
  call longdelay
  bcf portb, 6      ; Выключаем красный светодиод
  bsf portc, 4      ; Разматывание X-лебедки
  bcf portc, 6
  movlw 07h         ; Задержка на 1,4 с
  call longdelay    ; Для подтверждения маркера
  bsf status, 6     ; Страница 2
againr
  btfss cm1con0, 6  ; Разряд устанавливается, если напряжение
                    ;   на входе > 0,53 х напряжение питания
  goto againr
  bcf status, 6     ; Возврат к странице 0
  btfss flags, 0    ; Не останавливать?
  bcf portc, 4      ; Останов
  return
```

При вводе этого листинга с клавиатуры просто скопируйте листинг подпрограммы `left` и отредактируйте его.

### Перемещение Y-рамы

Эта рама также перемещает X-раму. Для этого используются две подпрограммы: `back` и `forward`, — которые перемещают Y-раму, переключая Y-лебедку, M1. Структура подпрограмм `back` и `forward` аналогична структуре подпрограмм `left` и `right` (листинг 10.5 и рис. 10.46).

**Листинг 10.5. Подпрограммы для перемещения Y-рамы вперед и назад**

```
back
  bsf portb, 6      ; Включаем красный светодиод
  movlw 03h
  call longdelay
  bcf portb, 6      ; Выключаем красный светодиод
  bsf portc, 4      ; Разматывание Y-лебедки
  bsf portc, 6
  movlw 07h         ; Задержка на 1,4 с
  call longdelay    ; Для подтверждения маркера
  bsf status, 6     ; Страница 2
againb
  btfss cm2con0, 6  ; Разряд устанавливается, если напряжение
                    ;   на входе > 0,53 х напряжение питания
  goto againr
  bcf status, 6     ; Возврат к странице 0
  btfss flags, 0
```



**Листинг 10.5. Окончание**

```
  bcf portc, 5      ; Останов
  return

forward
  bsf portb, 6      ; Включаем красный светодиод
  movlw 03h
  call longdelay
  bcf portb, 6      ; Выключаем красный светодиод
  bsf portc, 4      ; Наматывание Y-лебедки
  bsf portc, 6
  movlw 07h         ; Задержка на 1,4 с
  call longdelay    ; Для подтверждения маркера
  bsf status, 6     ; Страница 2
againf
  btfss cm2con0, 6  ; Разряд устанавливается, если напряжение
                    ;   на входе > 0,53 х напряжение питания
  goto againr
  bcf status, 6     ; Возврат к странице 0
  btfss flags, 0
  bcf portc, 5      ; Останов
  return
```

### Выход в требуемую точку

Четыре одношаговых подпрограммы выводят X-раму в любую точку рабочей области. Следующий фрагмент кода выводит ее в точку (0, 0), расположенной в переднем правом углу робота.

```
  bcf flags, 0      ; Активизация останова в подпрограммах
  call left         ; К самому правому столбцу
  call back         ; К переднему ряду
```

Начиная с базовой позиции, в которой рама была оставлена после выполнения подпрограммы из листинга 10.2, она должна пройти один шаг влево и один шаг назад. Когда для смещения рамы на один шаг используются подпрограммы перемещения, перед их вызовом необходимо установить или обнулить только разряд <0> регистра `flags`. Так, для перехода из базовой позиции в точку (0, 0), обнулите этот разряд, а затем вызовите подпрограммы `left` и `back`.

Подводя общий итог, отметим, что главная программа содержит:

- директивы, включая объявление меток регистров для хранения данных;
- раздел инициализации;
- подпрограмму перемещения X-рамы в базовую позицию;
- подпрограмму перемещения Y-рамы в точку (0, 0);

- подпрограммы `left`, `right`, `back`, `forward`, `delay` и `longdelay`;
- директиву `end`.

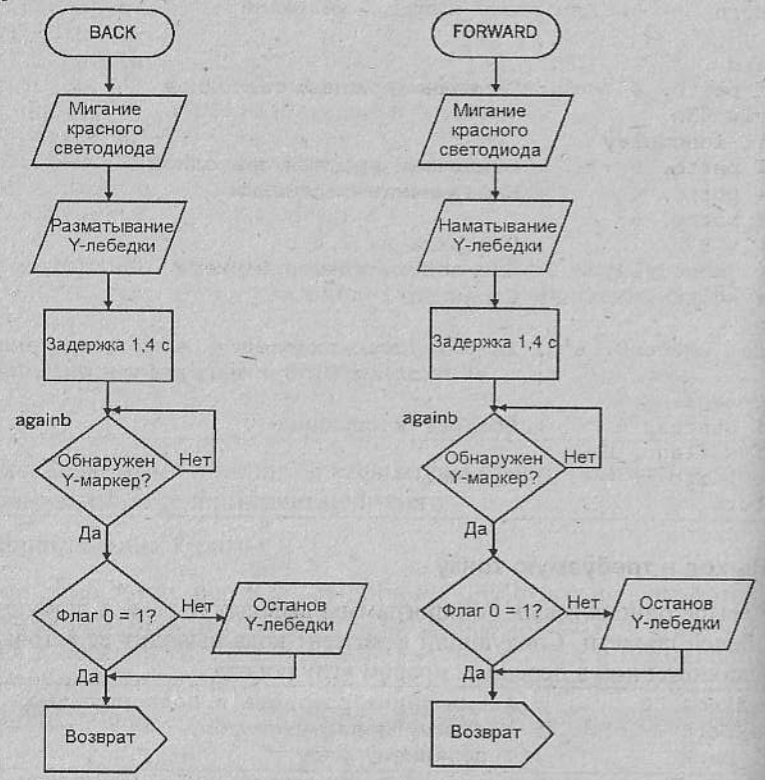



**Рис. 10.46.** Блок-схемы подпрограмм `back` (слева) и `forward` (справа)

Полезным добавлением к вышеперечисленному будет подпрограмма `flash`, которая обеспечивает неопределенно долгое мигание зеленого светодиода (листинг 10.6). Она расположена в конце главной программы, перед подпрограммами.

**Листинг 10.6. Подпрограмма, реализующая мигание зеленого светодиода**

```
flash
  bsf portb, 4        ; Включаем зеленый светодиод
  call delay
  bcf portb, 4        ; Выключаем зеленый светодиод
  call delay
  goto flash
```



Программа переходит прямо к подпрограмме `flash` после перемещения X-рамы в точку (0, 0). По мере поэтапного ввода и тестирования программы подпрограмма `flash` всегда остается в ее конце. Мигание светодиодов не является обязательным, однако всегда хорошо иметь визуальное подтверждение того, что микроконтроллер завершил выполнение задачи. Если светодиоды мигают, однако микроконтроллер *не* делает то, что ожидается, значит в программе присутствует ошибка, которую необходимо откорректировать.

### Перемещение из точки A в точку B

Простейший примером перемещения из точки A в точку B — подпрограмма перемещения из исходного положения в точку (0, 0). Для этого требуется всего лишь один шаг влево с последующим одним шагом назад. Теперь предположим, что X-рама уже находится в точке (0, 0), готовая к некоторому действию, предполагающему перемещение к центру рабочей области. Для этого ей, возможно, потребуется отработать два или большее число шагов влево и назад. Соответствующие подпрограммы должны вызываться для каждого шага.

В листинге 10.7 показана подпрограмма, перемещающая X-раму из точки (0, 0) в точку (3, 3).

**Листинг 10.7. Перемещение X-рамы из точки (0, 0) в точку (3, 3)**

```
; Перемещение в точку (3, 3)
  movlw 03h           ; Установка счетчика на 3
  movwf xpos
  movwf ypos
  bsf flags, 0        ; Блокирование остановки в подпрограмме
nextleft
  call left           ; В крайний правый столбец
  decfsz xpos, f      ; Обратный отсчет до нуля
  goto nextleft
  bcf portc, 4        ; Останов X-лебедки
nextback
  call back           ; В передний ряд
  decfsz ypos         ; Обратный отсчет до нуля
  goto nextback
  bcf portc, 5        ; Останов Y-лебедки
```

Данная подпрограмма использует для подсчета шагов два регистра: xpos и ypos. Для повторения вызовов по три раза в эти регистры записывается значение 3, а затем выполняется обратный отсчет до нуля.

Пример такой программы можно найти на прилагаемом к книге компакт-диске в папке `Портальный` (файлы `Gantry01.asm` и `Gantry01.hex`).

### Перемещение из точки C в точку D

Программа AtoB полезна при позиционировании X-рамы в начал сеанса. Она предполагает, что рама находится в позиции (0, 0), и пер водит ее в другую требуемую для работы точку. Однако, предположим что в какой-либо момент времени раме необходимо изменить координа ту. Один из способов реализовать это — вызвать подпрограмму возвр та рамы в базовую позицию, затем перевести ее в точку (0, 0), и тольк после этого — в требуемое положение. Но это займет много времени Гораздо быстрее перейти из одной точки непосредственно к другой б вывода рамы в базовую позиции и в точку (0, 0). В этом и заключает суть программы CtoD: перемещение из текущей точки C в точку D.

Данная подпрограмма требует шести регистров: xpos, ypos, xc, y xd и yd. Регистры xpos и ypos нам уже знакомы. Новые для нас регистр текущей координаты X (xc), регистр текущей координаты (yc), регистр координаты X точки назначения (xd) и регистр координ ты Y точки назначения (yd). Требуемые значения загружаются в пе перед вызовом подпрограммы. В данном случае мы предполагаем, что X-рама находится в позиции (5, 4) и должна переместится в точку (3, 1) Соответствующий фрагмент кода имеет вид:

```
movlw 05h    ; Текущее значение X в xc
movwf xc
movlw 04h    ; Текущее значение Y в yc
movwf yc
movlw 03h    ; Координата X точки назначения в xd.
movwf xd
movlw 01h    ; Координата Y точки назначения в yd
movwf yd
call xctod   ; Для перемещения влево или вправо
call yctod   ; Для перемещения вперед или назад
```

Блок-схема подпрограммы xctod (рис. 10.47) показывает расчет расстояния и направления по стартовой и конечной позициям. Вычита ние xc из xd дает расстояние, на которое необходимо переместить раму Если результат положителен, то X-рама перемещается влево через вы зов подпрограммы left.

Ситуация немного усложнится, если xc находится справа от xd, т.е значение xpos будет отрицательным. Подпрограммы left и right ра ботают только с положительными значениями, поэтому перед вызовом подпрограммы right значение xpos необходимо сделать положитель ным. Отрицательный результат вычитания приводит к установке флага переноса (STATUS <0>).



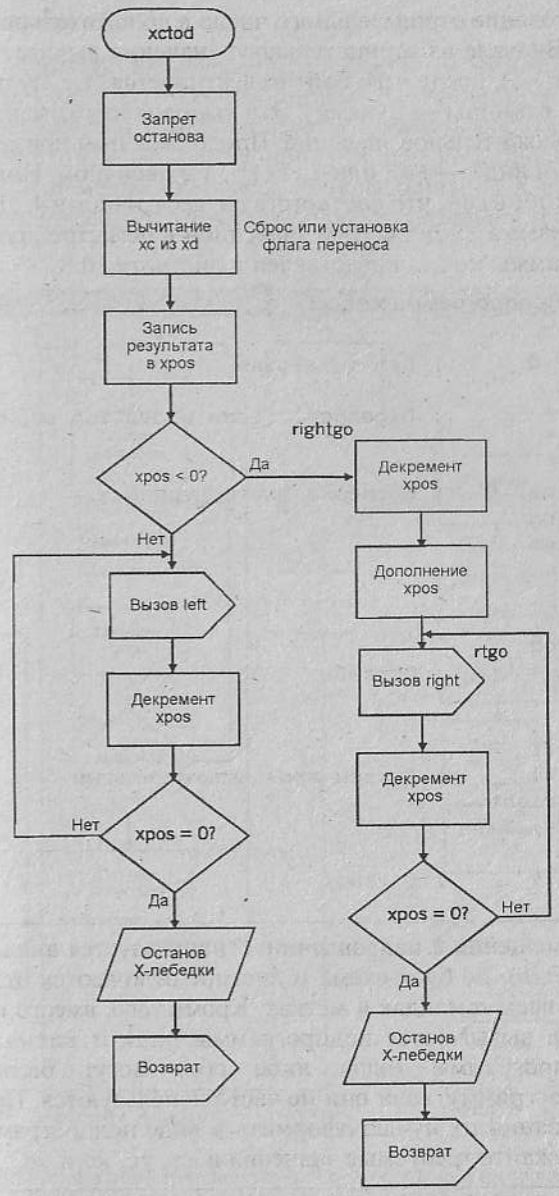



Рис. 10.47. Подпрограмма CtoD определяет направление и расстояние от текущей позиции, а затем перемещает X-раму непосредственно в новую точку

Преобразование отрицательного числа в положительное происходит в два этапа. Вначале из отрицательного значения вычитается 1 (например, –4 станет –5), после чего байт инвертируется (т.е. нули заменяются единицами, а единицы — нулями). Это выполняется командой `comf`. Результат — положительное значение. Продолжая наш пример, –5 в шестнадцатеричном виде — `FAh` или `11111011` в двоичном. Инверсией этого числа будет `00000100`, что соответствует десятичному 4. Далее вызывается подпрограмма `right` с этим значением в регистре `xpos`. Исходный код подпрограммы `xctod` представлен в листинге 10.8.

**Листинг 10.8. Подпрограмма `xctod`**

```
xctod
  bsf флаги, 0      ; Без остановки
  movf xc, w
  subwf xd, f       ; Перенос=1, если результат отрицателен
  movf xd, w
  movwf xpos
  btfsc status, 0   ; Проверка флага переноса
  goto lefttogo
  goto rightgo
lefttogo
  call left
  decfsz xpos, f
  goto lefttogo
  bcf portc, 4      ; Останов
  return
rightgo
  decf xpos, f
  comf xpos, f      ; Делаем xpos положительным
  rtgo call right
  decfsz xpos, f
  goto rtgo
  bcf portc, 4      ; Останов
  return
```

Для перемещения в направлении Y используется аналогичная подпрограмма `yctod`. Ее блок-схема и листинг отличаются от `xctod` заменой `x` на `y` во всех командах и метках. Кроме того, вместо подпрограмм `left` и `right` вызываются подпрограммы `back` и `forward`. Рассмотренные подпрограммы (одна либо обе) могут быть включены в основную программу, если они не часто используются. При интенсивном использовании их лучше оформить в виде подпрограмм. Перед их вызовом загружайте требуемые значения в `xc`, `yc`, `xd` и `yd`.

Пример такой программы можно найти на прилагаемом к книге компакт-диске в папке `Портальный` (файлы `Gantry02.asm` и `Gantry02.hex`).



### Сканирование

Одна из операций повышенной сложности — это сканирование рабочей области. Сканирование можно брать за основу при прохождении лабиринта для загрузки плана лабиринта в память робота. Затем робот анализирует этот план, решает лабиринт, и, наконец, прослеживает правильный путь, перемещая указатель, прикрепленный к X-раме.

Сканирование состоит из четырех последовательностей (рис. 10.48).

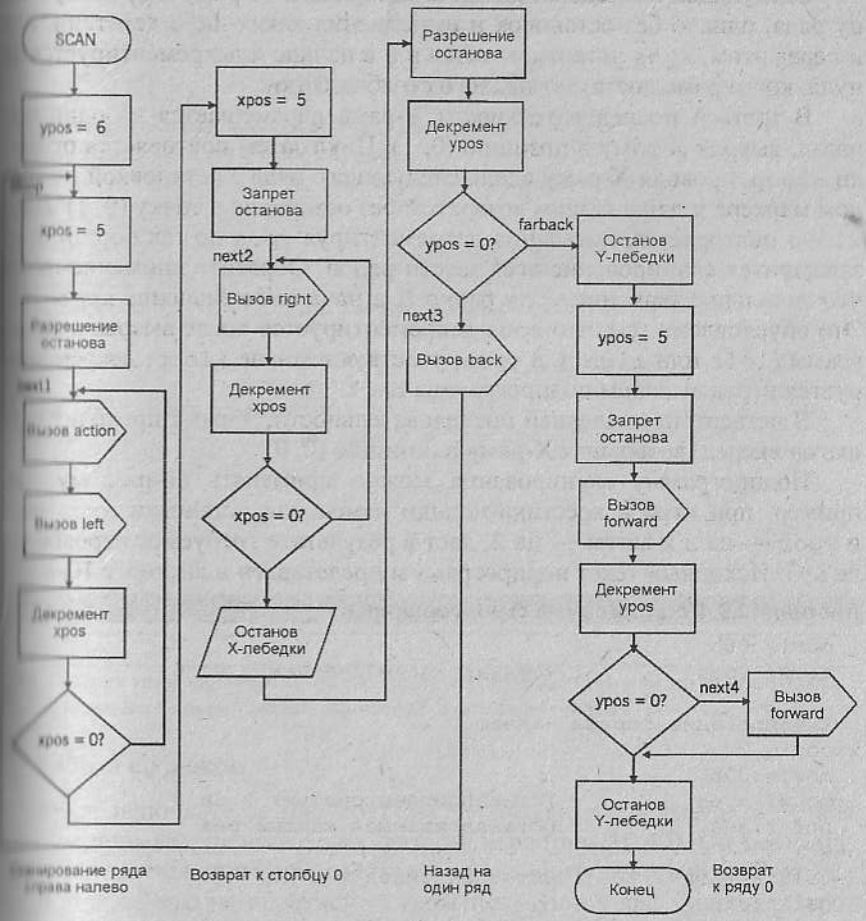



**Рис. 10.48.** Программа сканирования всей рабочей области и выполнения действия (например, распознавание объекта) в каждой позиции

Начиная от точки (0, 0), X-рама перемещается справа налево вдоль ряда 0. останавливаясь каждый раз при достижении маркера, включая первый в позиции (0, 0), и выполняя некоторое действие. В тестовой версии данной программы просто подается звуковой сигнал. После выхода из подпрограммы `left` декрементируется значение `xpos`. Этот регистр сордержит 5 в начале, поэтому ко времени, когда рама достигнет позиции (6, 0), он равен нулю.

Следующая последовательность возвращает X-раму к правому концу ряда, однако без остановок и выполнения каких-либо действий. Как и перед этим, `xpos` устанавливается в 5 в начале и декрементируется до нуля, когда рама достигает правого столбца сетки.

В третьей последовательности Y-рама перемещается на один шаг назад, выводя X-раму в позицию (0, 1). Цикл затем повторяется от метки `xloop`, проводя X-раму вдоль следующего ряда с остановкой на каждом маркере и дальнейшим возвратом без остановок в точку (0, 1). Цикл `xloop` повторяется, постоянно декрементируя `ypos` до тех пор, пока не завершится сканирование всех шести рядов. Обратите внимание на то, что начальное значение `ypos` равно 6, а начальное значение `xpos` — 5. Это обусловлено тем, что `xpos` декрементируется после вызова подпрограмм (`left` или `right`), а `ypos`, участвуя в цикле `xloop`, декрементируется перед вызовом подпрограммы `back`.

В четвертой, последней последовательности, Y-рама проходит пять шагов вперед, возвращая X-раму в позицию (0, 0).

Подпрограмму сканирования можно применять по-разному. Например, при игре в крестики-нолики изменение установки `xpos` на 2 и `ypos` — на 3 и затем — на 2, даст в результате требуемое игровое поле 3×3. Исходный текст подпрограммы представлен в листинге 10.9.

**Листинг 10.9. Подпрограмма сканирования**

```
  movlw 06h
  movwf ypos        ; Устанавливаем счетчик y на 6

; Сканирование справа налево
xloop
  movlw 05h
  movwf xpos        ; Устанавливаем счетчик x на 5
  bcf flags, 0      ; Останавливаемся каждый раз
next1
  call action       ; Опрос датчиков и т.п.
  call left
  decfsz xpos, f    ; Счетчик x  = 0?
  goto next1        ; Нет, продолжаем движение влево
  bcf portc, 4      ; Останов
```



**Листинг 10.9. Окончание**

```
  call action

; Возврат слева направо
  movlw 05h         ; Счетчик в 5
  movwf xpos
  bsf flags, 0      ; Без остановок
next2
  call right
  decfsz xpos, f    ; До обнаружения 5 маркеров
  goto next2
  bcf portc, 4      ; Останов

; Смещение Y-рамы на один ряд назад
  bcf flags, 0      ; останов.
  decfsz ypos, f    ; До обнаружения 5 маркеров
  goto next3
  goto farback
next3
  call back
  goto xloop        ; Для сканирования следующего ряда
farback
  bcf portc, 5      ; Останов

; Возврат Y-рамы в передний ряд
  movlw 05h         ; Счетчик y в 5
  movwf ypos
  bsf flags, 0      ; Без остановок
next4
  call forward
  decfsz ypos, f
  goto next4
  bcf portc, 5      ; Последняя остановка
```

Пример такой программы можно найти на прилагаемом к книге компакт-диске в папке `Портальный` (файлы `Gantry03.asm` и `Gantry05.asm`).

### Работа с крюком

Крюк используется для подъема и опускания предметов или, как в нашем примере, перемещения фигур в игре (рис. 10.49). Он поднимается и опускается двигателем M3, установленном у левого края Y-рамы.

Для обеспечения обратной связи используются два микропереключателя. Переключатель MS3 замыкается, когда шкивный блок полностью поднят и давит на его рычажок. Это дает нам известную позицию

крюка. Функция MS3 — та же, что и у двух микропереключателей на главной раме, которые срабатывают, когда X-рама находится в базовой позиции. Как только крюк полностью поднят, он может быть опущен на любую требуемую высоту включением двигателя на заданный период времени. Значения, используемые в соответствующих подпрограммах, настраиваются с учетом скорости вращения вала, передаточного числа коробки передач и диаметра катушки.

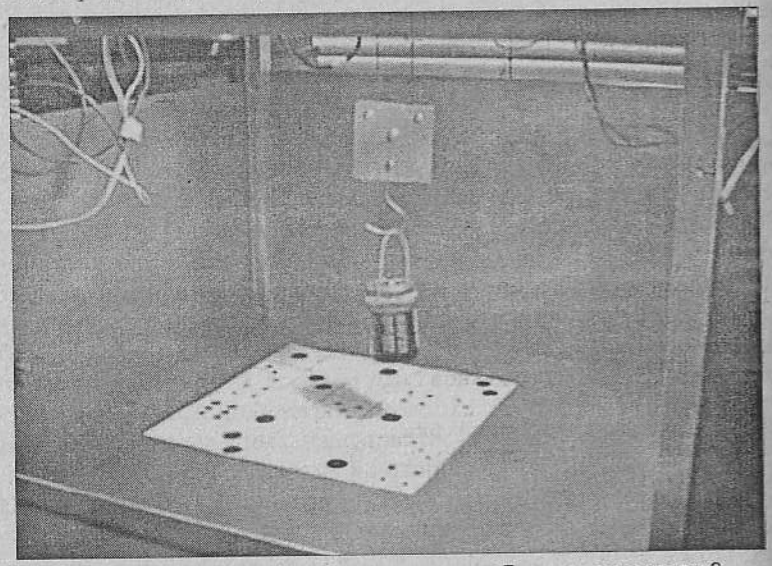


**Рис. 10.49.** Портальный робот играет в кости. Где крюк опустит груз?

Другой микропереключатель используется для того, чтобы распознавать наличие груза на крюку. Соответствующий механизм оснащен пружиной, натяжение которой необходимо отрегулировать таким образом, чтобы после поднятия крюком груза рычаг уходил вниз и замыкал ключ.

Оба переключателя подключены таким образом, что при своем срабатывании они заземляют входной вывод. Они соединены проводами с цифровыми входными каналами со слабыми подтягивающими сопротивлениями. Для микропереключателя полного подъема мы использовали межплатную линию X2, которая подсоединена к каналу RA0 (см. табл. 10.1). Для переключателя обнаружения груза мы использовали линию X1, которая идет к каналу RB5. Для обеспечения требуемых соединений три провода, идущие от крюка, подключаются к плате микроконтроллера PIC2 следующим образом:



- 0 В — на один из выводов 0 В (N7, N11 или N17);
- переключатель полного подъема — на X2 (J13);
- переключатель обнаружения груза — на X1 (I15).

Каждый из названных выводов соединяется напрямую со вторым выводом на той же полоске. Линии X1 и X2, идущие к плате микроконтроллера PIC1, подключаются ко второму выводу. Линия X0 при работе с крюком не используется, однако может быть задействована, например, для фотоэлемента.

Программа управления крюком представлена в листинге 10.10.

**Листинг 10.10. Программа управления крюком**

```
  movlw 0fh          ; Счетчик на 15
  movwf attempts
  call raise         ; Подъем крюка до предела

; Перемещение в точку (2, 3),

  movlw 02h          ; Установка счетчика на 2, 3
  movwf xpos
  movlw 03h
  movwf ypos
  bsf flags, 0       ; Запрет остановок в подпрограмме
nextleft
  call left          ; В крайний правый столбец
  decfsz xpos, f     ; Декрементирование до нуля
  goto nextleft
  bcf portc, 4       ; Останов X-лебедки
nextback
  call back          ; В передний ряд
  decfsz ypos, f     ; Декрементирование до нуля
  goto nextback
  bcf portc, 5       ; Останов Y-лебедки

; Обнаружение груза
  bcf flags, 0       ; Разрешение остановки
  bcf flags, 1       ; Крюк вниз
  call updown
findit
  call updown
  call left
  bsf flags, 1       ; Крюк вверх
  call updown
  call updown
  call delay         ; Время на установку.
  btfss portb, 5     ; Груз обнаружен?
  goto found
```

**Листинг 10.10. Окончание**

```
  call right
  decfsz attempts, f
  goto trymore
  goto abandon
trymore
  bcf flags, 1       ; Крюк вниз
  call updown
  call updown
  goto findit
found
  call raise

; Перемещение в позицию (5, 5) (из 3, 3)
  movlw 02h          ; Установ счетчиков на 2, 2
  movwf xpos
  movlw 02h
  movwf ypos
  bsf flags, 0       ; Запрет останова в подпрограмме
nextlft
  call left          ; Крайний правый столбец
  decfsz xpos, f     ; Декрементирование до нуля
  goto nextlft
  bcf portc, 4       ; Останов X-лебедки
nextbak
  call back          ; В задний ряд
  decfsz ypos, f     ; Декрементирование до нуля
  goto nextbak
  bcf portc, 5       ; Останов Y-лебедки

; Размещение груза
  call lower
  bcf flags, 0       ; Разрешение останова
abandon
  call right
  call right
  callraise
flash
  bsf portb, 4       ; Включение зеленого светодиода
  call delay
  bcf portb, 4       ; Выключение зеленого светодиода
  call delay
  goto flash
```

X-рама перемещается в точку (2, 3), где крюк цепляет груз. Затем он перемещается в позицию (5, 5) и опускает груз. В качестве груза может использоваться любой подходящий объект соответствующего веса (достаточного для срабатывания микропереключателя) с петлей или ручкой, за которую крюк мог бы зацепиться.



Программа разбита на несколько подпрограмм. Некоторые из них уже были описаны ранее: перемещение в базовую позицию (см. листинг 10.2) и `AtoB` (см. листинг 10.7). Подпрограмма `AtoB` вызывает подпрограммы `left`, `right`, `back` и `forward`, поэтому они также должны быть включены в файл с исходным кодом.

При работе с крюком используются следующие три подпрограммы:

- `raise` — поднимает крюк на максимальную высоту;
- `lower` — опускает крюк до момента соприкосновения с землей;
- `updown` — поднимает или опускает крюк в течении 0,2 с или другого предварительно заданного периода времени.

Подпрограммы `raise` и `lower` вызывают подпрограмму `updown`, которая в свою очередь вызывает подпрограмму `delay`. Будет ли подпрограмма `updown` поднимать или опускать крюк, зависит от состояния флага, который должен быть установлен перед ее вызовом. Этот флаг — разряд <1> регистра флагов. При этом нулю соответствует опускание, а единице — подъем.

В первой версии подпрограммы `raise` крюк просто поднимался до замыкания переключателя MS3. К сожалению, давление крюка на нижний блок вызывало срабатывание MS4, в силу чего микроконтроллер начинал действовать так, как будто на крюке есть груз. Для предотвращения этого одие вызов подпрограммы `updown` опускает крюк на один шаг после его подъема.

Данная программа начинается, как обычно, с инициализации микроконтроллера. В случае с крюком здесь есть одно отличие. Для банка 1 вместо `movlw 00h` используется команда `movlw 020h`, которая устанавливает вывод RB5 как вход. Кроме того, к списку меток добавлены `flags EQU 25h` и `attempts EQU 26h`.

Следующая секция программы перемещает X-раму в ее базовую позицию, а затем — в точку (0, 0), после чего программа работает согласно блок-схеме на рис. 10.50.

Подпрограмма перемещения в точку (2, 3) — такая же, как и подпрограмма из листинга 10.7, однако в ней `xpos` устанавливается на `02h`, а `ypos` — на `03h`.

На этом этапе программа входит в подпрограмму поиска груза. Предполагается, что он размещен в позиции (2, 3), однако его высота неизвестна. Крюк при этом выполняет последовательность движений, иллюстрируемых рис. 10.51. Он пытается зацепить петлю или ручку на грузе, двигаясь по замкнутой траектории справа, после чего поднимает-

ся вверх. Если груз не обнаружен, то крюк возвращается в правую точку, а затем опускается вниз на один шаг перед тем, как предпринять новую попытку. Он повторяет этот цикл до тех пор, пока после подъема не распознает наличие нагрузки на крюк.

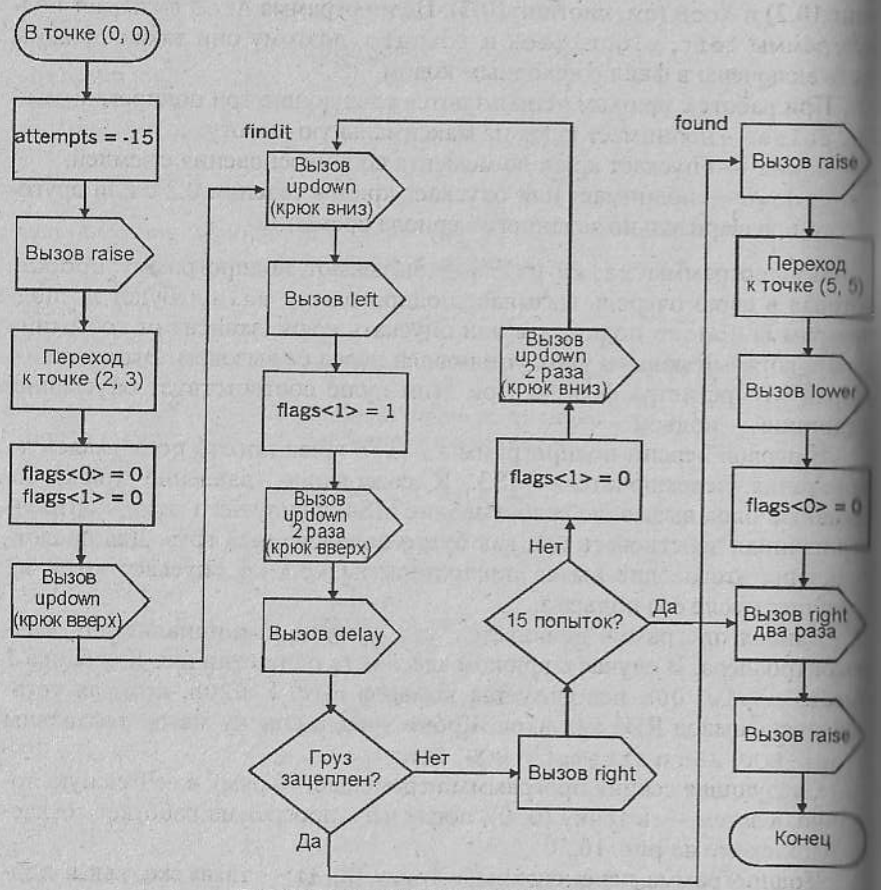



**Рис. 10.50.** Демонстрация действия крюка

Крюку предоставляется 15 попыток. Если он все же не сможет найти груз, то предполагается, что что-то не так (возможно, он прошел мимо груза, или груза нет вообще). В этом случае происходит переход в конец программы, рама отъезжает, крюк поднимается, и робот останавливается.



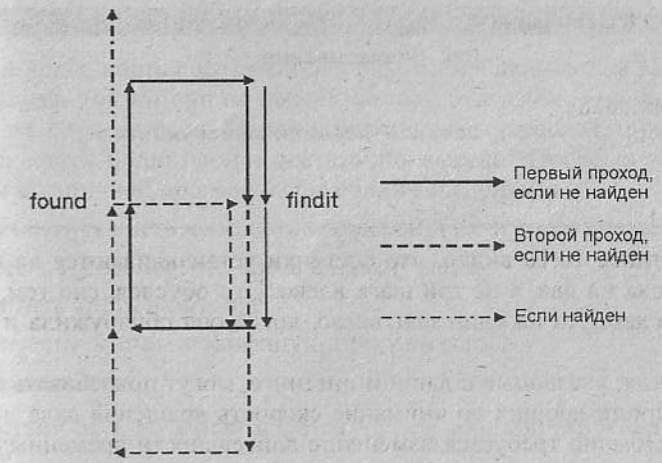



**Рис. 10.51.** Попытки крюка зацепить груз при поиске объекта неизвестной высоты

Листинг 10.10 завершает главную программу. В конце программы вызывается подпрограмма `flash` для индикации того, что микроконтроллер выполнил поставленную перед ним задачу. В качестве подпрограмм используются `delay`, `longdelay`, `left`, `right`, `back` и `forward`. Специальные завершающие подпрограммы показаны в листинге 10.11.

**Листинг 10.11. Специальные подпрограммы**

```
raise
  bsf flags, 1        ; Установка = подъем
  btfss porta, 0      ; MS3 замкнут?
  goto relax          ; Крюк вверх до предела
  call updown
  goto raise
relax
  bcf flags, 1
  call updown         ; Немного вниз
  return
lower
  bcf flags, 1        ; Сброс = вниз
  btfsc portb, 5      ; MS4 разомкнут? 0 = есть груз
  return              ; Груза нет или груз выпущен
  call updown
  goto lower
updown
  bsf portc, 3        ; Включаем лебедку инструмента
  bcf portc, 6        ; Наматывание троса
  btfss flags, 1
```

**Листинг 10.11. Окончание**

```
  bsf portc, 6      ; Или разматывание
  movlw 05h
  call longdelay
  bcf portc, 3      ; Останов лебедки инструмента
  return

end
```

В листинге 10.10 видно, что счетчики устанавливаются на перемещение крюка на два, а не три шага влево. Это обусловлено тем, что X-рама уже сдвинута на один шаг влево, когда она обнаружила и поднимает груз.

Значения, указанные в данном листинге, могут потребовать корректировки, принимающих во внимание скорость вращения вала и другие факторы. Обычно требуется изменение длительности временных задержек в подпрограммах `delay` и `longdelay` — особенно в подпрограмме поиска груза. Предполагается, что груз никогда не будет высоким. Может потребоваться опускание крюка в течении нескольких секунд перед тем, как он начнет поиск. Это означает, что максимальное количество попыток будет уменьшено.

Пример такой программы можно найти на прилагаемом к книге компакт-диске в папке `Портальный` (файлы `Gantry04.asm` и `Gantry04.hex`).

### Другие задачи для крюка

Объединение программы, подобной описанной выше, с подпрограммой генерирования случайных чисел может послужить основой для игры в кости (см. рис. 10.49). Игровое поле, например, может представлять собой квадратный кусок картона, размеченный квадратами в пяти строках и пяти столбцах. Крюк поднимает груз в фиксированной позиции и ставит его на один из квадратов. Игроки выигрывают или проигрывают в соответствии с правилами игры, которые принимаются заранее. Можно добавить увлекательности к этой игре, запрограммировав портального робота таким образом, чтобы он несколько раз "передумывал" перед тем, как фактически опустить груз.

Используя рассмотренную выше версию подрограммы сканирования, можно запрограммировать крюк на систематический просмотр рабочей области на наличие объектов с тем, чтобы переносить их в пустой ряд в дальней части рабочей области.



### Программирование кисти

Как и крюк, кисть поднимается и опускается двигателем M3, однако она оснащена только одним концевым выключателем. Он замыкается, когда кисть опускается на бумагу или в емкость с краской, и размыкается, когда кисть поднимается достаточно высоко. Последовательность действий в типичной программе рисования следующая.

1. Переместить кисть к емкости с краской и окунуть ее в краску.
2. Поднять и переместить кисть к началу мазка.
3. Опустить кисть на бумагу.
4. Переместить кисть в конечную точку мазка.
5. Поднять кисть.

Действия от 1 до 5 повторяются до тех пор, пока картина не будет завершена.

На этапах 1, 2 и 4 используются подпрограммы `xctod` и `yctod`. Для того чтобы запрограммировать рисование картины, все что необходимо — это справочная таблица последовательных значений `xd` и `yd`. Для текущего этапа в последовательности `xc` и `yc` — это значения, находящиеся в `xd` и `yd` после предыдущего этапа. Справочная таблица строится тем же образом, что и таблица для генерирования звуков (см. главу 9). Программа рисования несколькими цветами использует две или три емкости с краской плюс большую емкость с водой для промывки кисти при смене цвета.

Рисование картин и чертежей — это не единственные задачи для кисти. Она также может использоваться и в игровых программах (например, игре в крестики-нолики). Для вырисовывания крестов и нулей используйте тонкую кисть. Переместите кисть в выбранный квадрат с помощью подпрограмм `xctod` и `yctod`, а затем вызовите соответствующую подпрограмму рисования в квадрате.

### Программирование лазера

Лазер может работать как простая указка. В настольных играх, например, он может указывать фигуру, которую робот хочет переместить. Другое его применение связано со сканированием рабочей области. При прохождении лабиринта портальный робот использует лазер для сканирования плана лабиринта с его сохранением в памяти. Затем он логически решает задачу, и с помощью лазерного луча указывает путь выхода из лабиринта.

Лазер также может работать совместно с камерой.

### Программирование камеры

Камера содержит фоторезистор, включенный как делитель напря жения. Он может использоваться для измерения уровня яркости лазер ного луча, отраженного от объекта (игровая фигура, игровая доска, пла лабиринта) в рабочей области. Вместо лазера в качестве источника ос вещения может использоваться один либо большее число светодиодов а также окружающее освещение комнаты. При использовании камеры карта или другой целевой объект лучше всего распознается, когда он удерживается на уровне 60 мм под объективом камеры, всегда в фокусе

Для различения между белым (или светлым тоном) и черным (или темным тоном), выходно сигнал цепи датчика подается непосредствен но на компаратор 1 микроконтроллера PIC2 (RA1, вывод 18). Регистр CM1CON0 конфигурируется согласно описанию в главе 4. Регистр VRCON устанавливается на уровень, отличающий черную область от белой.

Для различения нескольких цветов (по их яркости) выходной сиг нал подается на один из 12 каналов АЦП микроконтроллера PIC1 или PIC2. Соединения платы PIC2 со входами PIC1 при этом следующие: X0 на AN4 (вывод 16); X1 на AN11 (вывод 12) и X2 на AN0 (вывод 19).

Для настройки схемы используйте тестер для измерения выходного напряжения при различаемых рабочих условиях. В случае применения компаратора используйте полученные замеры для вычисления опорного напряжения для VRCON, которое должно быть посередине между дву мя уровнями. В программе считывайте разряд CM1CON0<6>, который должен сожержать 0 для черного и 1 для белого цвета.

При использовании АЦП на основании полученных замеров вычис ляйте аналоговые значения, отличающие различные цвета. Реализуйте подпрограмму ветвления "Если A > B", которая на основании получен ных значений будет обрабатывать результаты съема данных в процессе работы программы.

После небольших экспериментов можно научиться различать чер ный и белый цвета, а также один или два оттенка серого (или други цвета, наподобие красного и зеленого). Эта способность полезна в игро вых программах при идентификации игровых фигур, принадлежащи двум игрокам (человек и робот), а также черных и белых квадратов на доске.

### Программирование схвата

Схват поднимается и опускается тем же механизмом, что и крюк поэтому программирование его движения вверх/вниз — аналогично



включая обратную связь от двух концевых выключателей. Кроме того, двигатель схвата должен контролироваться сигналами включения/выключения и открытия/закрытия. Присутствует также концевой выключатель, который срабатывает, когда губки схвата сжимаются.

Один из подходов к программированию в этом случае — распределенная обработка информации. Микроконтроллер PIC1 управляет тремя двигателями лебедок и последовательностью работы. Второй микроконтроллер (PIC2) используется для управления двигателями губок в соответствии с указаниями от PIC1. Микроконтроллер PIC2 размещен на X-раме.

Функции линий X1 и X2 — те же, что и при работе с крюком. Линия X0 для крюка не используется, однако в случае со схватом она отвечает за координирование управления открытием и закрытием губки. Эта линия идет от RC0 (выод 16) платы PIC1 к RB4 (вывод 13) платы PIC2. Сконфигурировав каналы на попеременный ввод и вывод, эту линию можно использовать для двусторонней связи между микроконтроллерами PIC.

Для микроконтроллера PIC1 используется подпрограмма, посылающая импульс по линии X0, который дает указание микроконтроллеру PIC2 открыть губку, если она закрыта, или закрыть ее, если она открыта. Когда состояние губки изменилось на противоположное, микроконтроллер PIC2 посылает ответный импульс микроконтроллеру PIC1, информируя его о том, что поставленная задача выполнена. Более подробно эту последовательность описывает блок-схема на рис. 10.52.

Главная программа, из которой на рис. 10.52 показано только начало, дает указание микроконтроллеру PIC2 открыть или закрыть губку. Это все, что должен делать PIC2, при условии, что все звенья программы PIC1 привязаны к последовательности открыть-закрыть-открыть-закрыть и т.д. Программа PIC2 может работать в цикле, который открывает или закрывает губку схвата, после чего посылает ответный импульс всякий раз, когда перед этим он принял импульс от PIC1. В промежутках он просто ожидает поступления следующего импульса.

При наличии дополнительных датчиков микроконтроллер PIC2 может решать и другие задачи, однако при этом должно обеспечиваться попеременное выполнение задач закрытия/открытия губки схвата.

### Подпрограммы портального робота

В табл. 10.4 перечислены подпрограммы, используемые для перемещения X-рамы. Этот перечень предназначен для удобства разработки программ для портального робота.

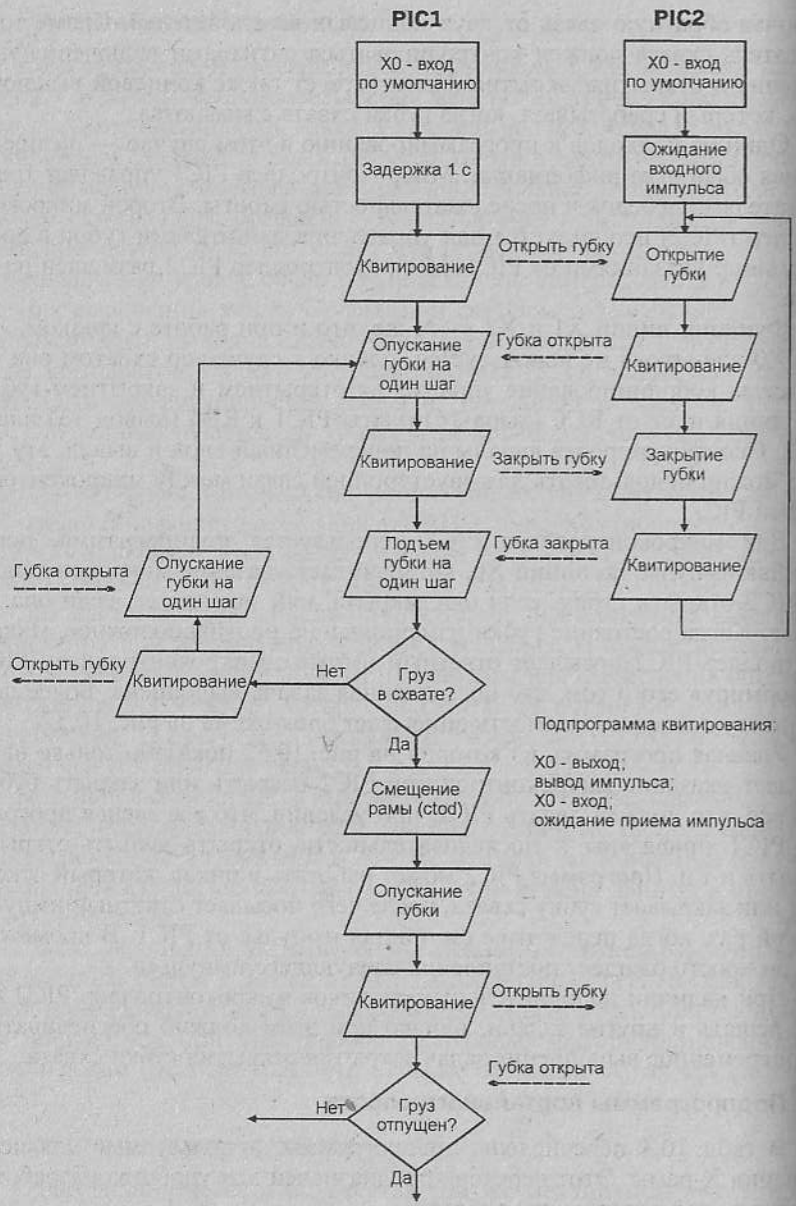



**Рис. 10.52.** Программы с квитированием связи, работающие одновременно на двух PIC



**Таблица 10.4.** Подпрограммы, используемые для перемещения X-рамы

| Имя | Листинг | До вызова | Действие | Вызываемые подпрограммы |
|---|---|---|---|---|
| `left` | 10.3 | `flags<0>` = 0 или 1 | Сдвиг влево на один шаг | `longdelay` |
| `right` | 10.4 | `flags<0>` = 0 или 1 | Сдвиг вправо на один шаг | `longdelay` |
| `back` | 10.5 | `flags<0>` = 0 или 1 | Сдвиг назад на один шаг | `longdelay` |
| `forward` | 10.5 | `flags<0>` = 0 или 1 | Сдвиг вперед на один шаг | `longdelay` |
| `atob` | 10.7 | `xpos`, `ypos` | Сдвиг на количество рядов и столбцов, заданных xpos и ypos | `left`, `right`, `back`, `forward` |
| `xctod` | 10.8 | `xc`, `xd`, `flags<0>` | Сдвиг из текущего X к целевому | `left`, `right`, `back`, `forward`, `longdelay` |
| `yctod` | 10.8 (преобразуйте xtod) | `yc`, `yd`, `flags<0>` | Сдвиг из текущего Y в целевое | `left`, `right`, `back`, `forward`, `longdelay` |
| `scan` | 10.9 | `xpos`, `ypos` | Сканирует справа налево, спереди назад, с остановкой справа и возвратом в начало | `left`, `right`, `back`, `forward` |
| `raise` | 10.11 | Нет | Подъем крюка вверх | `updown` |
| `lower` | 10.11 | Нет | Опускание крюка до снятия груза | `updown` |
| `updown` | 10.11 | `flags<1>` | Подъем или опускание 0,2 с | `delay` |

### Флаги

Разряды регистра `flags` оказывают следующий эффект:

- разряд <1>: 0 — крюк вниз; 1 — крюк вверх;
- разряд <0>: 0 — останов в конце подпрограммы `left`/`right`/`back`/`forward`; 1 — без остановки.

# Содержимое прилагаемого к книге компакт-диска

Компакт-диск содержит исходные тексты рассмотренных в книге программ и подпрограмм, разбитые на несколько папок:

- `Андроид` — программы для робота “Андроид” (см. главу 7);
- `Игрушка` — программы для робота-игрушки (см. главу 8);
- `Искатель` — программы для робота “Искатель” (см. главу 9);
- `Портальный` — программы для портального робота (см. главу 10);
- `Сегменты` — директивы настройки типичной программы, а также некоторые часто используемые подпрограммы;
- `Скутер` — программы для робота “Скутер” (см. главу 6).

## Издательство "МК-Пресс" представляет


Авраменко Ю.Ф.

### Мобильные телефоны LG. Ремонт и обслуживание. Том I (+CD)

ISBN 978-966-8806-29-2
576 стр., мягкая обложка

Книга составлена на основании сервисной документации LG Electronics. В ней подробно рассмотрены схемотехнические решения современных мобильных телефонов на базе БИС обработки аналоговых и цифровых сигналов производства ANALOG DEVICES. Представлены сведения о работе всех функциональных устройств телефонов стандарта GSM. На прилагаемом к книге компакт-диске, приводятся электрические принципиальные схемы на рассмотренные модели и справочные данные на элементную базу ведущих производителей интегральных схем для систем беспроводной связи. Книга рассчитана на широкий круг специалистов, занимающихся сервисным обслуживанием мобильных телефонов.


Авраменко Ю.Ф.

### Мобильные телефоны LG. Ремонт и обслуживание. Том II (+CD)

ISBN 978-966-8806-32-2
576 стр., мягкая обложка

Книга составлена на основании сервисной документации LG Electronics. В ней подробно рассмотрены схемотехнические решения современных мобильных телефонов на базе БИС обработки аналоговых и цифровых сигналов производства ANALOG DEVICES. Представлены сведения о работе всех функциональных устройств телефонов стандарта GSM. На прилагаемом к книге компакт-диске, приводятся электрические принципиальные схемы на рассмотренные модели, а также сервисная документация на мобильные телефоны производства SAMSUNG и NOKIA. Книга рассчитана на широкий круг специалистов, занимающихся сервисным обслуживанием мобильных телефонов.



## Издательство "МК-Пресс" представляет


Крид Хадлстон

### Проектирование интеллектуальных датчиков с помощью Microchip dsPIC (+CD)

ISBN 978-966-8806-38-4
320 стр., мягкая обложка

На страницах этой книги раскрыты способы применения популярных цифровых контроллеров сигналов Microchip dsPIC, в которых вычислительный потенциал мощных цифровых процессоров сигналов удачно объединен с возможностями микроконтроллеров PIC. Рассматриваются вопросы не только программирования, но и проектирования электронного оборудования. Таким образом, читатель получает полное представление о процессе создания интерфейса для трех конкретных типов датчиков: температуры, давления/нагрузки и расхода. Книга раскрывает реальные проблемы, возникающие в повседневной работе разработчиков, и показывает решения, позволяющие реализовать все сильные стороны интеллектуальных датчиков.


Авраменко Ю.Ф.

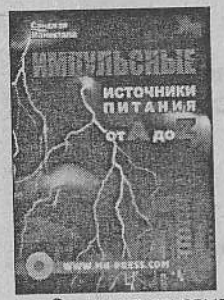
### Качественный звук — сегодня это просто

ISBN 966-8806-27-1
286 стр., мягкая обложка

В книге максимально подробно приведены все рекомендации разработчиков – инженеров NSC, как правильно построить усилительный тракт на основе мощных ОУ. Современный подход, основанный на рекомендациях инженеров AD и TI, к топологии печатной платы, к выбору «правильных» пассивных компонентов для звуковоспроизводящего тракта поможет реализовать основной принцип: как можно меньше ухудшить качество записи. Большое количество примеров построения качественных УМЗЧ будет наглядным пособием для реализации собственной конструкции в короткие сроки с небольшими материальными затратами и главное, с предсказуемым результатом.

Книги издательства "МК-Пресс" можно заказать:
по адресу: **02002, г.Киев, а/я 294**; по телефону/факсу: **(044) 517-73-77**,
по e-mail: **info@mk-press.com**
или приобрести в магазине "**Микроника**" по адресу: **г.Киев, ул. М.Расковой, 13**
Посетите наш Internet-магазин: **http://www.mk-press.com**

## Издательство "МК-Пресс" представляет

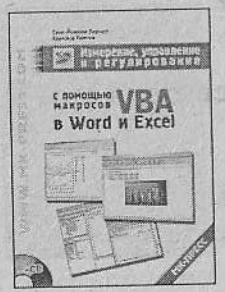



Санджая Маниктала

### Импульсные источники питания от A до Z (+CD)

ISBN 978-5-903383-59-7
544 стр., мягкая обложка

Эта книга основывается на десятилетнем авторском опыте проектирования источников питания. Здесь читатель найдет наглядное и доступное введение в курс "Источники питания"; изложение основ без пугающего математического анализа; полную и, в то же время, уникальную по своей простоте методику проектирования импульсных преобразователей и их магнитных компонентов; подробный расчет всех видов потерь в импульсных источниках питания; описание основных схемотехнических решений импульсных источников; исчерпывающее исследование аспектов контроля и измерения паразитных электромагнитных излучений, связанных с работой импульсных преобразователей.




Г.-Й. Берндт, Б. Каинка

### Измерение, управление и регулирование с помощью макросов VBA в Word и Excel (+CD)

ISBN 978-5-7931-0504-0
256 стр., мягкая обложка

Эта книга представляет новый подход, согласно которому весь диапазон задач измерения, управления и регулирования реализуется средствами популярного программного пакета Microsoft Office. Хотя это звучит необычно, с помощью приложений Word и Excel можно получить прямой доступ к аппаратному обеспечению, что делает их универсальными и простыми в использовании инструментами.

В книге показано, как с помощью макросов VBA реализовать управление цифровыми мультиметрами, релейными картами и ПК-интерфейсами, организовать взаимодействие с микроконтроллерными системами и многое другое на основе стандартного последовательного интерфейса RS232.



## Издательство "МК-Пресс" представляет


Авраменко Ю.Ф.

### Мощные биполярные транзисторы для импульсных источников питания, TV-приемников и мониторов

ISBN 966-8806-17-4
544 стр., мягкая обложка

В справочнике представлены электрические параметры на мощные биполярные транзисторы, имеющие высокую скорость переключения. Данные приборы применяются в импульсных источниках питания различного назначения, в промышленном оборудовании, в бытовой и профессиональной видео- и аудиотехнике. Указаны технические данные на изделия следующих ведущих производителей полупроводниковых приборов: FAIRCHILD, HITACHI, MOTOROLA (ON SEMICONDUCTOR), PANASONIC, PHILIPS, SANKEN, SAMSUNG, SANYO, SHINDENGEN, ST-MICROELECTRONICS, TOSHIBA и ZETEX. Таблица аналогов полупроводниковых приборов составлена на основании руководства Master Replacement Guide.


Тяпичев А.Г.

### Персональный компьютер в радиолюбительской практике

ISBN 966-8806-18-2
400 стр., мягкая обложка

Книга предназначена для любознательного читателя и любителей мастерить своими руками. В ней описаны различные варианты специального использования персонального компьютера для выполнения «нетрадиционных» для него работ, таких как управление различными удаленными аппаратами, кодирование и декодирование различных сигналов, создание принципиальных электрических схем и проверка работоспособности этих схем, создание звуковых эффектов и многое другое. Большое внимание уделено процессу программирования микроконтроллеров. Даны описания специальных компьютерных программ, и подробное описание аппаратов, которые могут подключаться к компьютеру и применяться в совместной с ним работе.

Книги издательства "МК-Пресс" можно заказать:
по адресу: **02002, г.Киев, а/я 294**; по телефону/факсу: **(044) 517-73-77**,
по e-mail: **info@mk-press.com**
или приобрести в магазине "**Микроника**" по адресу: **г.Киев, ул. М.Расковой, 13**
Посетите наш Internet-магазин: **http://www.mk-press.com**

ББК 32.973-04
УДК 004.312
Б67

**Бишоп О.**

Б67 Настольная книга разработчика роботов. – К.: "МК-Пресс", СПб.: "КОРОНА-ВЕК", 2010. – 400с., ил.

ISBN 978-5-7931-0546-0 ("КОРОНА-ВЕК")
ISBN 978-966-8806-64-3 ("МК-Пресс")
ISBN 978-0-7506-6556-8 (англ.)

Эта книга представляет собой справочное руководство для тех, кто хочет научиться проектировать и конструировать роботов. Благодаря представленным в ней пошаговым инструкциям, вы быстро освоите методики создания забавных и захватывающих роботов. На основании своего обширного практического опыта автор открывает важные аспекты программирования, электроники и механики, характерные для робототехники. Поскольку все проекты основаны на использовании всемирно популярных микроконтроллеров PIC, методики программирования осваиваются быстро и безболезненно.

Данное руководство — идеальный вариант для новичков в сфере робототехники. Оно будет также полезно тем опытным разработчикам, которые хотят расширить свои познания в области программирования роботов. И безусловно, эта книга пригодится студентам, выполняющим практические задания по проектированию систем на основе микроконтроллеров.

**ББК 32.973-04**

*Главный редактор: Ю. А. Шпак*

Подписано в печать 25.09.2009. Формат 60 × 84 1/16. Бумага газетная. Печать офсетная.
Усл. печ. л. 23,3. Уч.-изд. л. 18,2. Тираж 2000 экз. Заказ №884

СПД Савченко Л.А., Украина, г. Киев, тел./факс: (044) 517-73-77; e-mail: info@mk-press.com.
Свидетельство о внесении субъекта издательского дела в Государственный реестр
издателей, производителей и распространителей издательской продукции:
серия ДК №51582 от 28.11.2003г.



ISBN 978-5-7931-0546-0 ("КОРОНА-ВЕК")
ISBN 978-966-8806-64-3 ("МК-Пресс")
ISBN 978-0-7506-6556-8 (англ.)

# Настольная книга разработчика РОБОТОВ

**Откройте для себя мир робототехники в ходе разработки и программирования собственных роботов**

**Пошаговые инструкции с полным описанием процесса проектирования пяти роботов**

- Каждая модель разработана и протестирована лично автором
- Все этапы сопровождаются крупными фотографиями и доступными инструкциями
- Рассмотренные проекты являются превосходной практической базой в освоении ключевых методик разработки электронной и механической части роботов, а также их программирования.

Эта книга представляет собой справочное руководство для тех, кто хочет научиться проектировать и конструировать роботов. Благодаря представленным Оуэном Бишопом пошаговым инструкциям, вы быстро освоите методики создания забавных и захватывающих роботов. На основании своего обширного практического опыта Оуэн открывает важные аспекты программирования, электроники и механики, характерные для робототехники. Поскольку все проекты основаны на использовании всемирно популярных микроконтроллеров PIC, методики программирования осваиваются быстро и безболезненно.

Данное руководство — идеальный вариант для новичков в сфере робототехники. Оно будет также полезно тем опытным разработчикам, которые хотят расширить свои познания в области программирования роботов. И безусловно, эта книга пригодится студентам, выполняющим практические задания по проектированию систем на основе микроконтроллеров.

Компакт-диск содержит исходные коды программ для рассмотренных в книге роботов.