

# Содержание

---

Благодарности .....	14
Введение .....	15
<hr/>	
<b>I Основы оборудования .....</b>	<b>16</b>
Введение .....	17
<b>Глава 1. Основы электроники .....</b>	<b>19</b>
Цепи постоянного тока .....	19
Напряжение и сила тока .....	19
Резисторы .....	21
Электрические цепи .....	24
Мощность .....	27
Цепи переменного тока .....	28
Конденсаторы .....	29
Индуктивности .....	34
Активные компоненты .....	35
Собираем все элементы вместе – источник питания .....	39
Осциллограф .....	43
Средства управления .....	43
Зонды .....	46
<b>Глава 2. Логические цепи .....</b>	<b>49</b>
Кодирование чисел .....	49
Двоично-десятичное представление .....	52
Комбинаторная логика .....	52
Логический элемент НЕ .....	53
Логические элементы И и НЕ-И .....	53
Логические элементы ИЛИ и НЕ-ИЛИ .....	55
Исключающее ИЛИ .....	55
Схемы .....	56
Устройства с тремя состояниями .....	59
Последовательная логика .....	59
Логическое резюме .....	64
<b>Глава 3. Советы по разработке аппаратных средств .....</b>	<b>65</b>
Диагностика .....	65
Средства подключения .....	66
Другие рекомендации .....	67
Резюме .....	69

<b>II</b>	<b>Проектирование</b> .....	<b>70</b>
	<b>Введение</b> .....	<b>71</b>
	<b>Глава 4. Инструментальные средства и методы улучшения качества программного кода</b> .....	<b>73</b>
	Введение .....	73
	Традиционный цикл последовательной разработки встраиваемой системы .....	73
	Типичные проблемы современного рынка встраиваемых систем .....	74
	Общие методы повышения качества кода и сокращения сроков выхода на рынок .....	75
	Фиксируйте спецификацию и работайте параллельно .....	75
	Создавайте контрольные отметки .....	75
	Используйте доступные ресурсы .....	75
	Непрерывно обучайте своих сотрудников .....	76
	Основные факторы, влияющие на продолжительность цикла разработки .....	76
	Какой этап длится дольше других? .....	77
	Как снизить время разработки ПО и повысить качество кода .....	77
	Пишите код в соответствии с внутренним руководством по оформлению ПО .....	77
	Выполняйте проверку кода .....	79
	Выбирайте подходящие инструментальные средства разработки .....	79
	Повторное использование вместо повторного изобретения .....	82
	Как сократить сроки проектирования аппаратной части системы .....	83
	Используйте как можно больше готовой продукции .....	83
	Тщательно подбирайте микроконтроллер .....	83
	Пример микроконтроллеров, сокращающих сроки выхода на рынок, – микроконтроллеры Philips .....	84
	Резюме и перспективы .....	84
	<b>Глава 5. Советы по улучшению функций</b> .....	<b>86</b>
	Минимизируйте функциональные возможности .....	86
	Инкапсулируйте .....	87
	Избавляйтесь от избыточности .....	88
	Сокращайте код реального времени .....	88
	Ход программы должен быть грациозным .....	89
	Беспощадно улучшайте программы .....	89
	Применяйте стандарты и экспертизу .....	90
	Тщательно комментируйте программу .....	91
	Резюме .....	93
	<b>Глава 6. Эволюционная разработка</b> .....	<b>95</b>
	Введение .....	95
	1. История .....	96

2. Проблемы, решаемые методом Эво .....	97
А. Парадоксы требований .....	97
В. Очень короткие циклы.....	98
С. Быстрая и частая обратная связь .....	100
D. Фиксация сроков.....	101
Е. Оценка, планирование и контроль.....	103
F. Разница между напряженной работой и выполнением заказа .....	104
G. Обязательства.....	106
H. Риски .....	107
I. Производственные совещания .....	107
J. Волшебные слова .....	108
3. Как мы используем метод Эво в работе над проектом.....	109
А. День Эво.....	109
В. Последний день цикла .....	110
С. Производственное совещание.....	111
4. Памятки .....	112
А. Критерии назначения приоритетов заданиям.....	112
В. Критерии назначения приоритетных сроков завершения промежуточных этапов .....	112
С. Критерии завершения задания .....	113
5. Использование метода Эво в новых проектах .....	114
6. Тестирование в методе Эво.....	116
7. Запросы об изменениях и отчеты о проблемах .....	117
8. Инструментальные средства.....	117
9. Выводы.....	119
Благодарности .....	121
Ссылки .....	121
<b>Глава 7. Реализация встраиваемого конечного автомата .....</b>	<b>123</b>
Конечные автоматы .....	123
Пример .....	124
Реализация.....	127
Тестирование.....	130
Запуск системы .....	131
Ссылки .....	131
<b>Глава 8. Иерархические конечные автоматы.....</b>	<b>132</b>
Пример традиционного конечного автомата .....	133
Пример иерархического конечного автомата.....	135
<b>Глава 9. Разработка приложений, критически важных для обеспечения безопасности.....</b>	<b>140</b>
Введение .....	140
Надежность и безопасность .....	141
История документа DO-178B.....	141
Обзор стандарта DO-178B .....	142

Классификация неисправных состояний .....	143
Анализ архитектуры системы .....	144
Разбиение на разделы .....	144
Несколько версий разнородного ПО .....	145
Мониторинг безопасности .....	145
Документация по архитектуре системы .....	146
Жизненный цикл программного обеспечения согласно стандарту DO-178B .....	146
Планирование.....	146
Разработка .....	147
Процесс разработки.....	147
Виды деятельности при разработке ПО .....	148
Верификация требований к ПО.....	148
Верификация проектирования ПО .....	149
Верификация программного кода.....	149
Верификация процесса интеграции .....	149
Верификация процесса верификации .....	149
Управление конфигурацией.....	150
Обеспечение качества ПО (SQA).....	151
Технология объектно-ориентированного программирования и проблемы приложений, критически важных для обеспечения безопасности.....	151
Итеративный процесс .....	152
Проблемы сертификации объектно-ориентированных приложений.....	152
Автоматическая генерация кода .....	153
Автоматическая генерация тестов .....	155
Возможность оперативного контроля .....	155
Управление конфигурацией.....	155
Структурный охват .....	156
Невыполняемые/деактивированные участки программы.....	156
Наследование и множественное наследование .....	156
Резюме.....	157
Ссылки .....	157

## **Глава 10. Установка и использование системы контроля версий ..... 158**

Введение .....	158
Мощь и элегантность простоты.....	159
Контроль версий .....	160
Типичные признаки отказа от использования (неполного использования) системы контроля версий .....	160
Простые системы контроля версий.....	161
Усовершенствованные системы контроля версий .....	161
Для каких файлов нужно использовать контроль версий .....	162
Совместная работа с файлами и клиенты системы контроля версий.....	162
Нет локального клиента, нет общей файловой системы.....	163
Нет локального клиента, но есть общая файловая система.....	163

Есть локальный клиент, но нет общей файловой системы .....	163
Есть и локальный клиент, и общая файловая система .....	163
Проблемы интегрированной среды разработки .....	164
Проблемы графического интерфейса пользователя .....	164
Спецификация SCC .....	165
Интерфейс для веб-браузера или клиент-Java-систем контроля версий .....	165
Основные положения концепции контроля версий .....	166
Советы.....	171
Отслеживание ошибок .....	174
Неконфигурационные средства управления .....	176
ПО для зеркального отображения информации.....	176
Автоматизированное резервное копирование .....	176
Веб-браузер .....	176
Группы новостей в Интернете .....	177
Заключительные комментарии .....	177
Рекомендованная литература, ссылки и ресурсы .....	178

---

## **III Математика** ..... 180 |

### **Введение** ..... 181 |

### **Глава 11. Введение в машинные вычисления**..... 182 |

Введение .....	182
Целочисленная арифметика .....	182
Деление и отрицательные числа .....	182
Целые типы и их размер .....	185
Переполнение или исчезновение значащих разрядов.....	186
Математические операции с плавающей запятой.....	189
Неожиданный результат.....	189
Форматы с плавающей запятой .....	190
Погрешности округления.....	192
Ошибки при умножении и делении .....	194
Ошибки при сложении и вычитании .....	195
Обработка ошибок при вычислениях с плавающей запятой.....	197
Использование эквивалентных выражений для устранения катастрофической потери точности .....	199
Арифметические операции с фиксированной запятой.....	201
Область применимости.....	201
Представление чисел с фиксированной запятой и операции над ними .....	202
Обработка ошибок при выполнении операций с фиксированной запятой.....	203
Заключение.....	204
Библиография.....	204

<b>Глава 12. Аппроксимации для вычислений с плавающей запятой</b> .....	205
Общие замечания о тригонометрических функциях.....	206
Косинус и синус.....	207
Более точное вычисление косинуса.....	213
Тангенс.....	214
Более точное вычисление тангенса.....	219
Арктангенс, арксинус и арккосинус.....	220
<b>Глава 13. Математические функции</b> .....	224
Код Грея .....	224
Умножение целого на константу .....	224
Вычисление исключаяющего ИЛИ.....	224
Извлечение квадратного корня в целых числах.....	225
Важнейшие математические функции.....	225
<b>Глава 14. Стандарт IEEE 754 для чисел с плавающей запятой</b> .....	226
Специальные значения.....	227
<hr/>	
<b>IV Системы реального времени</b> .....	229
<b>Введение</b> .....	230
<b>Глава 15. Ядра реального времени</b> .....	231
Введение .....	231
Что такое ядро реального времени?.....	231
Что такое задача?.....	232
Тактовый интервал таймера.....	235
Планирование задач .....	236
Переключение контекстов.....	238
Службы ядра .....	239
Службы ядра. Семафоры.....	239
Службы ядра. Очереди сообщений.....	243
Службы ядра. Управление памятью.....	245
Нужно ли вам ядро?.....	245
Можете ли вы использовать ядро? .....	246
Выбор ядра .....	247
Заключение.....	250
<b>Глава 16. Реентерабельность</b> .....	251
Атомарные переменные.....	251
Еще два правила.....	253
Обеспечение реентерабельности кода.....	254
Рекурсия .....	256

Асинхронность оборудования/микропрограммного обеспечения .....	257
Состояние конкуренции .....	258
Варианты решения проблемы .....	259
Другие ОС реального времени .....	261
Метастабильные состояния .....	262
Микрокод, а не оборудование .....	264
<b>Глава 17. Латентность прерываний .....</b>	<b>268</b>
Получение данных .....	271
<b>Глава 18. Как работает ваш компилятор языка C: минимизация размеров программы.....</b>	<b>274</b>
Современные компиляторы языка C.....	275
Структура компилятора .....	275
Смысл программы .....	277
Базовые преобразования .....	277
Распределение регистров .....	279
Вызовы функций.....	280
Подстановка функций.....	280
Сжатие кода низкого уровня.....	281
Компоновщик.....	281
Управление оптимизацией, осуществляемой компилятором .....	282
Модель памяти.....	283
Советы по программированию .....	283
Правильно подбирайте размер переменных.....	283
Используйте указатели наиболее подходящего типа.....	284
Структуры и байты заполнения.....	285
Используйте прототипы функций.....	286
Используйте параметры .....	287
Не используйте операцию получения адреса .....	287
Не используйте встроенный ассемблер .....	288
Не пишите остроумный код .....	288
Проверяйте значения битовых полей перед использованием .....	290
Следите за использованием библиотечных функций.....	290
Используйте дополнительные подсказки компилятору .....	291
Финальные замечания .....	291
Благодарности .....	292
<b>Глава 19. Оптимизация кода на языках C и C++ .....</b>	<b>293</b>
Устанавливайте размеры структур равными степени двойки .....	293
Размещайте метки «case» как можно ближе друг к другу.....	293
Размещайте наиболее используемые метки case вначале .....	293
Разбивайте крупные операторы switch на вложенные операторы-переключатели .....	294
Минимизируйте число локальных переменных .....	295

Описывайте локальные переменные как можно глубже внутри функций .....	295
Сокращайте число аргументов .....	295
Используйте ссылки при передаче параметров и возвращаемого значения для типов, имеющих длину более 4 байтов .....	296
Не определяйте возвращаемое значение, если оно не используется.....	296
Учитывайте расположение ссылок относительно кода и данных .....	296
Старайтесь использовать тип <code>int</code> вместо <code>char</code> или <code>short</code> .....	297
Пишите облегченные конструкторы .....	298
Старайтесь использовать инициализацию вместо присваивания .....	298
Используйте списки инициализации конструкторов .....	299
Не объявляйте функции виртуальными «на всякий случай» .....	299
Используйте подстановку для функций длиной в 1–3 строки .....	299
<b>Глава 20. Макросы <code>assert</code> в системах реального времени .....</b>	<b>301</b>
Проблемы встраиваемых систем.....	301
Макросы <code>assert</code> в системах реального времени .....	303
<hr/>	
<b>V Ошибки и исправления.....</b>	<b>309</b>
<b>Введение .....</b>	<b>310</b>
<b>Глава 21. Реализация загружаемого микрокода с помощью флеш-памяти.....</b>	<b>311</b>
Введение .....	311
Микропрограмматор .....	312
Преимущества микропрограмматоров.....	312
Недостатки микропрограмматоров.....	313
Получение микропрограмматора.....	313
Базовый микропрограмматор .....	314
Типичные проблемы и их решение .....	316
Отладчику «не нравятся» перезаписываемые области памяти .....	316
Отладчикам «не нравится» код, выполняющий перемещение самого себя.....	317
Невозможность генерации позиционно-независимого кода.....	319
Отсутствие микрокода в момент загрузки .....	320
Постоянная блокировка по времени .....	320
Неожиданное отключение питания.....	321
Аппаратные альтернативы.....	322
Разделение кода и данных .....	323
Гибкость и надежность.....	323
<b>Глава 22. Диагностика памяти.....</b>	<b>325</b>
Тестирование ПЗУ .....	325
Тестирование ОЗУ .....	327



<b>Глава 23. Энергонезависимая память</b> .....	333
Контролирующие схемы.....	333
Запись многобайтных значений.....	335
Тестирование.....	339
Выводы.....	340
<b>Глава 24. Профилактическая отладка</b> .....	341
Стеки и кучи.....	341
Заполнение памяти.....	344
Блуждающий код.....	346
Специальные дешифраторы.....	348
Блоки управления памятью.....	349
Выводы.....	350
<b>Глава 25. Обработка исключительных ситуаций на C++</b> .....	351
Горы (ориентиры безопасности исключительных ситуаций).....	352
История этой территории.....	353
Коварная ловушка.....	354
Смола!.....	356
Самый легкий путь.....	357
Оператор присваивания – специальный случай.....	359
В плохую погоду.....	360
Подведем итоги.....	363
Литература.....	367
<b>Глава 26. Отличный сторожевой таймер</b> .....	368
Внутренние сторожевые таймеры.....	371
Внешние сторожевые таймеры.....	374
Характеристики отличных сторожевых таймеров.....	375
Использование встроенного сторожевого таймера.....	379
Внешний сторожевой таймер.....	381
Сторожевые таймеры для многозадачной среды.....	383
Выводы и некоторые соображения.....	385
<b>Приложение А. ASCII-коды</b> .....	388
<b>Приложение Б. Выравнивание и порядок байтов</b> .....	390
Ограничения, накладываемые на выравнивание байтов.....	390
Для чего нужно ограничивать выравнивание байтов?.....	390
Общие правила выравнивания байтов.....	392
Выравнивание структур для повышения эффективности.....	392
Порядок байтов.....	393
Почему используется различный порядок байтов?.....	393
Подпрограммы преобразования.....	393
<b>Указатель</b> .....	395

# Благодарности

---

Я хотел бы поблагодарить всех авторов, чьи материалы вошли в состав данной книги. Это был огромный труд, но наконец-то все готово!

Также спасибо Кэрол Льюис, редактору издательства Newnes по новым проектам, и ее мужу Джеку, которые терроризировали меня целый год, пока я не пришел к такой концепции книги, которая, как мне кажется, должна быть понятна и читателям, и авторам. Кэрол положила начало моей авторской карьере более десяти лет назад, подвигнув меня на написание книги «*The Art of Programming Embedded Systems*» («Искусство программирования встраиваемых систем»). Спасибо ей за участие на протяжении всех этих лет.

И особая благодарность – моей жене Мэрибет за помощь и поддержку, а также за колоссальный объем работы по форматированию материалов и административному руководству проектом.

– Джек Гансл,  
пристань Анкоридж,  
г. Балтимор, штат Мэриленд

# Введение

---

Вот оно – «Руководство по микропрограммному обеспечению»! Эта книга заполняет важнейший пробел в литературе по встраиваемому ПО. Существует настоятельная потребность в сборнике идей и концепций, справочнике, настольной книге инженеров, куда они заглядывали бы, чтобы найти решение своих задач и освежить в памяти забытый материал. Главной темой этой книги является микрокод, однако суровая реальность мира встраиваемого ПО такова, что код и аппаратура взаимозависимы. Они не могут существовать в изоляции; ни в одной другой области программирования нет такой глубокой связи между реальным и виртуальным.

Аналоговые инженеры постоянно твердят, что у них прекрасная профессия. Конечно, очень здорово ворочать операционными усилителями. Но бедняги не ведают, как это увлекательно – сделать так, чтобы все двигалось, огоньки мигали, газ тек. Это мы, разработчики встраиваемого ПО, управляем работой моторов, перекачиваем кровь, приводим в действие автомобильные тормоза, контролируем развертку телевизионного изображения по горизонтали и по вертикали и выдвигаем компакт-диски из дисководов. Что может сравниться по притягательности с этой размытой границей между микрокодом и реальным миром?

Книга адресована разработчикам микрокода, пишущим те самые программы, на которых работают технологии XXI века. Эта книга создавалась не как учебник или вводный курс по написанию микрокода. Существует множество других пособий, цель которых – научить людей азам разработки встраиваемых систем. Не является она и введением в основы программирования. Каждый разработчик должен знать, что такое конструктивная стоимостная модель (СОСОМО) Бёма, используемая для оценки затрат на разработку ПО; методы экстремального программирования; подходы Фагена и Гильба к инспектированию программного обеспечения; персональный метод разработки по Хамфри; модель оценки зрелости процессов разработки и сопровождения программного обеспечения, созданная в институте Software Engineering Institute, и ряд других хорошо известных и постоянно развивающихся методов. Тщательное программирование – ключевая составляющая успеха при создании любой большой системы, и мир встраиваемого ПО в этом не оригинален.

В данном руководстве очень мало информации по операционным системам реального времени, ТСП/IP, цифровым сигнальным процессорам и другим подобным темам. Эти актуальные вопросы крайне важны для огромного множества встраиваемых приложений. Однако для освещения каждого из таких вопросов требуется отдельное издание – и таких книг уже очень и очень много.

# I Основы оборудования

II	Проектирование	72
III	Математика	183
IV	Системы реального времени	232
V	Ошибки и исправления	312

- 21 Глава 1. Основы электроники
- 51 Глава 2. Логические цепи
- 67 Глава 3. Советы по разработке аппаратных средств

# Введение

Первые электронные вычислительные машины были аналоговыми, в действительности они представляли собой совокупность операционных усилителей и ничего более. Того, что мы сейчас называем «программами», вообще не существовало, вместо них разработчики алгоритмов использовали массивы электронных компонентов, помещавшихся в петли обратной связи таких усилителей. Программирование сводилось в буквальном смысле к переключению проводов в машине. Только инженеры, хорошо разбиравшиеся в электротехнике, могли управляться с подобными монстрами.

С появлением в 1940-х гг. цифровых компьютеров, способных записывать информацию, программы превратились из электрических схем в биты, хранящиеся на различных носителях... хотя это было совершенно не похоже на то, что мы имеем сегодня! Менее очевидным преимуществом стало абстрагирование программиста от машины. Цифровая сущность вычислительных машин преодолела ограничения электрических параметров – больше не существовало ничего, кроме нуля и единицы. Возможности машинных вычислений открылись для огромной группы людей, гораздо более широкой, чем специалисты по электротехнике. Программирование стало самостоятельной дисциплиной со своими профессиональными приемами, ни один из которых не требовал даже самых элементарных знаний электроники и схемотехники.

Единственное исключение – встраиваемые системы. Изобретение микропроцессора корпорацией Intel (1971 г.) вернуло компьютеры назад – к самым истокам. Вычислительные машины сразу стали достаточно доступными по цене и небольшими по размерам, чтобы их можно было встраивать в какую-либо продукцию. Несмотря на низкую стоимость процессора, снижение общей стоимости систем снова оказалось проблемой для проектировщиков и программистов новой интеллектуальной продукции.

Это последний рубеж вычислительной техники, на котором оборудование и микропрограммное обеспечение (микрокод, *firmware*) не отделимы друг от друга. Зачастую можно понизить стоимость системы, используя компоненты с более интеллектуальной программой, или повысить производительность, применив другой способ крепления. Наши микропрограммы тесно связаны с особенностями периферийного оборудования и датчиков, а также со множеством явлений реального мира. Хотя и существует тенденция приглашать на работу «чистых» программистов, самыми лучшими разработчиками всегда оказываются те, кто имеет не только богатый опыт в разработке программного обеспечения, но и практические навыки в области электроники.

Каждый разработчик встраиваемого ПО должен – нет, *просто обязан!* – знать содержание этой главы. Вы не имеете права заявить: «Ну, поскольку я программист-разработчик микропрограммного обеспечения, мне не нужно знать, что такое резистор». Ваша задача как разработчика *встраиваемого* ПО – создание системы, которая будет больше, чем просто программным кодом.

Более глубокие сведения по всем аспектам электроники содержатся в замечательной книге «*The Art of Electronics*» (русский перевод: Хоровиц П., Хилл У. Искусство схемотехники: в 2 т. М.: Мир, 1986).

Джек Ганссел – автор ежемесячной колонки «Breakpoints» в разделе «*Embedded Systems Programming*» электронного еженедельника «Embedded Pulse», выходя-

щего на веб-сайте [embedded.com](http://embedded.com); он написал четыре книги по встраиваемым системам и одну о своем неудачном опыте мореплавателя. Он начал заниматься встраиваемыми системами в начале 70-х гг. еще на процессоре 8008. С тех пор успел основать и продать три электронные компании, в том числе одно из крупнейших предприятий по созданию инструментальных средств для разработки встраиваемого ПО. Как программист и руководитель Джек принимал участие в работах по созданию более 100 встраиваемых продуктов: от глубоководного навигационного оборудования до системы безопасности Белого дома. В настоящее время он ведет семинары, посвященные наилучшим способам создания встраиваемых систем, для компаний по всему миру.

# Глава 1. ОСНОВЫ ЭЛЕКТРОНИКИ

Джек Ганссл

## Цепи постоянного тока

*Постоянный ток* (DC) – замечательный термин, обозначающий сигналы, не изменяющиеся со временем. Здесь отсутствуют пики, как на электроэнцефалограмме мертвого мозга или на выходе аккумуляторной батареи. Источник питания вашего ПК получает постоянный ток из переменного тока (AC) обычной электросети.

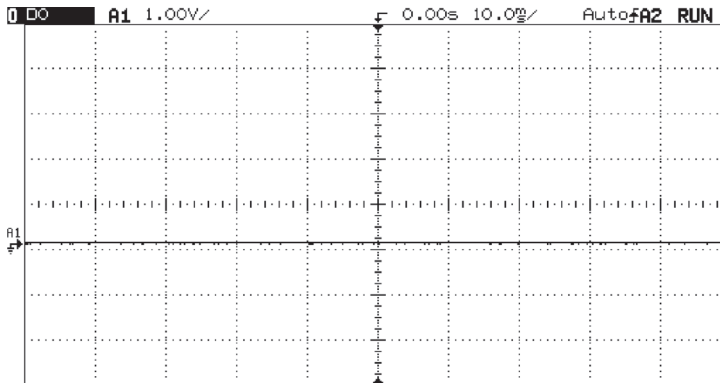


Рис. 1.1. Постоянный ток характеризуется сигналом постоянной, неменяющейся амплитуды

## Напряжение и сила тока

Для характеристики электрического тока обычно пользуются напряжением и силой тока, не задумываясь, что за обеими величинами стоят фундаментальные закономерности физики. Атомы, имеющие недостаток или избыток электронов, называются *ионами*. Ионы могут иметь как положительный, так и отрицательный заряд. Два иона с противоположным зарядом (один положительный, что означает нехватку у него электронов, а другой отрицательный, обладающий одним или более дополнительным электроном) притягивают друг друга. Сила такого притяжения называется электродвижущей силой и обычно обозначается как ЭДС<sup>1</sup>.

Заряды измеряются в кулонах (Кл), 1 кулон равен заряду  $6,25 \times 10^{18}$  электронов (для отрицательных зарядов) или протонов (для положительных).

Ампер (А) – это заряд в 1 Кл, протекающий через данную точку за одну секунду. Напряжение 1 В – это разность потенциалов, благодаря которой проводник с текущим по нему током в 1 ампер совершит работу в 1 джоуль. Джоуль в секунду называется ваттом.

<sup>1</sup> Вообще говоря, сила взаимодействия точечных зарядов называется кулоновской, а под термином ЭДС понимается физическая величина, равная отношению работы сторонних сил по перемещению заряда вдоль замкнутого контура к величине этого заряда. – *Прим. пер.*

Следует заметить, что на самом деле редкие специалисты по электротехнике помнят эти определения и практически никто ими не пользуется.



*Рис. 1.2. Даже старомодный недорогой аналоговый вольт-омметр, подобный изображенной на рисунке модели Radio Shack, измеряет постоянный ток, как минимум, не хуже осциллографа*

Старинная, но очень удачная аналогия представляет электрический ток как воду, текущую по трубе: в этом случае сила тока будет количеством воды, пропускаемой трубой в единицу времени, а напряжение – давлением воды. Хотя обычно сила тока измеряется в амперах, для компьютера значение силы тока 1 А является крайне высоким. Большинству цифровых и аналоговых цепей требуются гораздо меньшие токи. В табл. 1.1 приведен перечень наиболее употребительных единиц силы тока.

**Таблица 1.1. Единицы измерения силы тока**

Название	Сокращенное обозначение	Значение [А]	Где встречается
Ампер	А	1	Источники питания; наиболее высокопроизводительным процессорам может потребоваться ток силой во многие десятки ампер
Миллиампер	мА	0,001	Логические схемы, процессоры (десятки или сотни мА), обычные аналоговые схемы
Микроампер	мкА (μА)	10 <sup>-6</sup>	Экономичные логические и аналоговые схемы, ОЗУ с резервным питанием от батарей
Пикоампер	пкА	10 <sup>-12</sup>	Очень чувствительные аналоговые входные цепи
Фемтоампер	фА	10 <sup>-15</sup>	Измерение аналоговых сигналов с помощью новейших технологий

Для большинства встраиваемых систем характерен не столь экстремальный диапазон напряжений. Типичные источники питания для логических схем и микропроцессоров обеспечивают напряжение от одного–двух до пяти вольт. Напряжение источников питания аналоговых устройств редко выходит за преде-



лы диапазона плюс или минус 15 В. Некоторые аналоговые сигналы от датчиков могут достигать значений милливольтового (0,001 В) диапазона. Радиоприемники способны обнаруживать сигналы микровольтового уровня, однако для этого используются довольно сложные методы шумоподавления.

## Резисторы

Когда электроны движутся по проводам, внутри компонентов или – при несчастном случае – через тело человека, они встречают *сопротивление*, то есть стремление проводника ограничить ток электронов. Совершенным резистором является вакуум – через него электрический ток вообще не течет. Сходными характеристиками обладает и воздух, но влажный воздух может в некоторой степени проводить электрический ток, так как вода – достаточно хороший проводник.

Сверхпроводники, единственные материалы с нулевым сопротивлением, обладают этим свойством благодаря волшебной силе квантовой механики при крайне низких температурах – порядка температуры кипения азота или ниже. Остальные материалы, даже самые лучшие проводники, имеют сопротивление. Потрогайте шнур питания своего 1500-ваттного керамического нагревателя – он теплый, поскольку некоторое количество электрической мощности рассеивается в шнуре вследствие сопротивления провода.

Сопротивление измеряется в омах; чем больше эта величина, тем хуже проводник. Международное обозначение ома – большая греческая буква «омега» ( $\Omega$ ). Сопротивление, напряжение и сила тока связаны наиболее фундаментальным из всех законов электротехники – законом Ома:

$$V = I \times R,$$

где  $V$  – напряжение [В],  $I$  – сила тока [А], а  $R$  – сопротивление [Ом]. (Инженерам-электрикам нравится использовать букву «E» для обозначения напряжения, поскольку она может обозначать и электродвижущую силу).

Что это означает на практике? Подайте ток силой 1 А на нагрузку величиной 1 Ом и между ее контактами вы получите напряжение в 1 В. Увеличьте напряжение вдвое, и, если сопротивление осталось неизменным, сила тока удвоится. Хотя все электронные компоненты имеют сопротивление, *резисторы* используются специально для снижения проводимости. Они применяются везде. Регулятор громкости в стереосистеме (не в цифровой) – это сопротивление, значение которого меняется, когда вы вращаете ручку; при повышении сопротивления снижается уровень сигнала и, следовательно, уровень громкости динамиков.

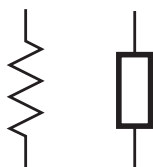


Рис. 1.3. Зигзагообразная линия слева – стандартное обозначение резистора на электрических схемах. Справа приведено обозначение, принятое в Великобритании. Как говаривал Черчилль, «мы – два народа, разделенных общим языком»

Таблица 1.2. Номиналы реальных резисторов

Название	Сокращенное обозначение	Значение [Ом]	Где встречается
Миллиом	мОм (mΩ)	0,001	Сопrotивление проводов и других хороших проводников
Ом	Ом (Ω)	1	В источниках питания используются большие понижающие сопротивления с номиналом от нескольких ом до нескольких десятков Ом
Сотни ом			Во встраиваемых системах довольно часто можно обнаружить резисторы с номинальным сопротивлением в несколько сотен ом, которые используются для ограничения высокоскоростных сигналов
Килоом	кОм (kΩ, или просто к)	1000	Резисторы с номинальным значением от половины кОм до сотни или более кОм можно встретить во всех электронных приборах. Типичные значения «нагрузки» составляют от нескольких до десятков кОм
Мегаом	МОм (MΩ)	10 <sup>6</sup>	Аналоговые цепи с сигналом низкого уровня
Сотни мегаом		10 <sup>8</sup> и более	Счетчики Гейгера и другие крайне чувствительные приборы; в резисторах встречается редко, поскольку близко по значению к сопротивлению воздуха

Что произойдет, если соединить несколько сопротивлений? Полное эффективное сопротивление системы последовательно соединенных резисторов будет равно сумме их значений:

$$R_{EFF} = R_1 + R_2.$$

Для двух резисторов, соединенных параллельно, эффективное сопротивление можно вычислить по формуле:

$$R_{EFF} = \frac{R_1 \times R_2}{R_1 + R_2}.$$

(Таким образом, два одинаковых резистора, соединенные параллельно, действуют как одно эффективное сопротивление, равное половине любого из них: если взять два сопротивления по 1 кОм, то эффективное сопротивление составит 500 Ом. Добавим третий резистор: это эквивалентно тому, что параллельно с резистором в 500 Ом подключен резистор 1 кОм, и эффективное сопротивление составит 333 Ом.)

Обобщенная формула для цепи, содержащей более двух параллельно соединенных резисторов:

$$R_{EFF} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4} + \dots}.$$

Для обозначения номинальных сопротивлений отдельных резисторов производители используют цветовую маркировку. Хотя на первый взгляд этот подход может показаться слишком загадочным, на практике он себя оправдывает – независимо от ориентации и способа установки резистора на печатной плате цветные полоски на нем всегда остаются видимыми.

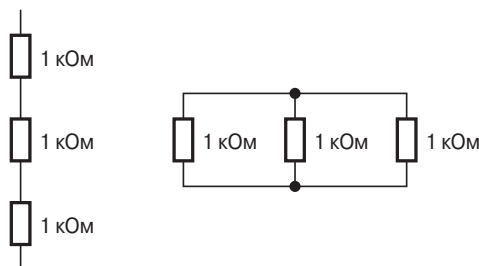


Рис. 1.4. Три последовательно соединенных резистора слева эквивалентны одному компоненту с сопротивлением 3000 Ом. Правые резисторы, соединенные параллельно, работают как одно сопротивление со значением 333 Ом

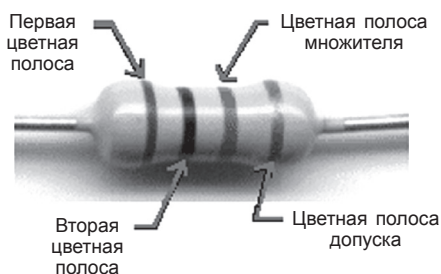


Рис. 1.5. На этой черно-белой фотографии цвет полосок не виден. Однако мы можем прочитать шифр слева направо: две первые описывают целую часть номинала, третья – множитель. Четвертая полоса может быть золотой (5%) или серебряной (10%), характеризуюя допуск значения

**Таблица 1.3. Цветовая маркировка резисторов. Все прежние мнемонические правила для запоминания порядка цветов теперь неприменимы. Единственная приемлемая, хотя и менее запоминающаяся, альтернативная фраза звучит так: «Big Brown Rabbits Often Yield Great Big Vocal Groans When Gingerly Slapped»<sup>1</sup>**

Цвет полосы	Значение	Множитель
Черный	0	1
Коричневый	1	10
Красный	2	100
Оранжевый	3	1000
Желтый	4	10 000
Зеленый	5	100 000
Синий	6	1 000 000
Фиолетовый	7	Не используется
Серый	8	Не используется
Белый	9	Не используется
Золотой (третья полоса)		÷10
Серебряный (третья полоса)		÷100

<sup>1</sup> На самом деле запомнить порядок цветов довольно легко: после черного и коричневого идут все цвета радуги, кроме голубого, дальше – серый и белый. – Прим. пер.

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

[e-Univers.ru](http://e-Univers.ru)