Содержание

Благодарности 15

Введение 16

Содержание CD-ROM 17

I Основы оборудования 18

Введение 19

Глава 1. Основы электроники 21

Цепи постоянного тока 21
Напряжение и сила тока 21
Резисторы 23
Электрические цепи 26
Мощность 29
Цепи переменного тока 30
Конденсаторы 31
Индуктивности 36
Активные компоненты 37
Собираем все элементы вместе – источник питания 41
Осциллограф 45
Средства управления 45
Зонды 48

Глава 2. Логические цепи 51

Кодирование чисел 51
Двоично-десятичное представление 54
Комбинаторная логика 54
Логический элемент НЕ 55
Логические элементы И и НЕ-И 55
Логические элементы ИЛИ и НЕ-ИЛИ 57
Исключающее ИЛИ 57
Схемы 58
Устройства с тремя состояниями 61
Последовательностная логика 61
Логическое резюме 66

Глава 3. Советы по разработке аппаратных средств 67

Диагностика 67
Средства подключения 68

Другие рекомендации 69
Резюме 71

II Проектирование 72

Введение 73

Глава 4. Инструментальные средства и методы улучшения качества программного кода 75

Введение 75
Традиционный цикл последовательной разработки встраиваемой системы 75
Типичные проблемы современного рынка встраиваемых систем 76
Общие методы повышения качества кода и сокращения сроков выхода на рынок 77
Фиксируйте спецификацию и работайте параллельно 77
Создавайте контрольные отметки 77
Используйте доступные ресурсы 77
Непрерывно обучайте своих сотрудников 78
Основные факторы, влияющие на продолжительность цикла разработки 78
Какой этап длится дольше других? 79
Как снизить время разработки ПО и повысить качество кода 79
Пишите код в соответствии с внутренним руководством по оформлению ПО 79
Выполняйте проверку кода 81
Выбирайте подходящие инструментальные средства разработки 81
Повторное использование вместо повторного изобретения 84
Как сократить сроки проектирования аппаратной части системы 85
Используйте как можно больше готовой продукции 85
Тщательно подбирайте микроконтроллер 85
Пример микроконтроллеров, сокращающих сроки выхода на рынок, – микроконтроллеры Philips 86
Резюме и перспективы 86

Глава 5. Советы по улучшению функций 88

Минимизируйте функциональные возможности 88
Инкапсулируйте 89
Избавляйтесь от избыточности 90
Сокращайте код реального времени 90
Ход программы должен быть грациозным 91
Беспощадно улучшайте программы 91
Применяйте стандарты и экспертизу 92
Тщательно комментируйте программу 93
Резюме 95

Глава 6. Эволюционная разработка 97
Введение 97
1. История 98
2. Проблемы, решаемые методом Эво 99
A. Парадоксы требований 99
B. Очень короткие циклы 100
C. Быстрая и частая обратная связь 102
D. Фиксация сроков 103
E. Оценка, планирование и контроль 105
F. Разница между напряженной работой и выполнением заказа 106
G. Обязательства 108
H. Риски 109
I. Производственные совещания 109
J. Волшебные слова 110
3. Как мы используем метод Эво в работе над проектом 111
A. День Эво 111
B. Последний день цикла 112
C. Производственное совещание 113
4. Памятки 114
A. Критерии назначения приоритетов заданиям 114
B. Критерии назначения приоритетных сроков завершения промежуточных этапов 114
C. Критерии завершения задания 115
5. Использование метода Эво в новых проектах 116
6. Тестирование в методе Эво 118
7. Запросы об изменениях и отчеты о проблемах 119
8. Инструментальные средства 119
9. Выводы 121
Благодарности 123
Ссылки 123

Глава 7. Реализация встраиваемого конечного автомата 125
Конечные автоматы 125
Пример 126
Реализация 129
Тестирование 132
Запуск системы 133
Ссылки 133

Глава 8. Иерархические конечные автоматы 134
Пример традиционного конечного автомата 135
Пример иерархического конечного автомата 137

Глава 9. Разработка приложений, критически важных для обеспечения безопасности 142
Введение 142

Надежность и безопасность ... 143
История документа DO-178B ... 143
Обзор стандарта DO-178B ... 144
Классификация неисправных состояний ... 145
Анализ архитектуры системы ... 146
Разбиение на разделы ... 146
Несколько версий разнородного ПО ... 147
Мониторинг безопасности ... 147
Документация по архитектуре системы ... 148
Жизненный цикл программного обеспечения согласно стандарту DO-178B ... 148
Планирование ... 148
Разработка ... 149
Процесс разработки ... 149
Виды деятельности при разработке ПО ... 150
Верификация требований к ПО ... 150
Верификация проектирования ПО ... 151
Верификация программного кода ... 151
Верификация процесса интеграции ... 151
Верификация процесса верификации ... 151
Управление конфигурацией ... 152
Обеспечение качества ПО (SQA) ... 153
Технология объектно-ориентированного программирования и проблемы приложений, критически важных для обеспечения безопасности ... 153
Итеративный процесс ... 154
Проблемы сертификации объектно-ориентированных приложений ... 154
Автоматическая генерация кода ... 155
Автоматическая генерация тестов ... 157
Возможность оперативного контроля ... 157
Управление конфигурацией ... 158
Структурный охват ... 158
Невыполняемые/деактивированные участки программы ... 158
Наследование и множественное наследование ... 159
Резюме ... 159
Ссылки ... 160

Глава 10. Установка и использование системы контроля версий ... 161

Введение ... 161
Мощь и элегантность простоты ... 162
Контроль версий ... 163
Типичные признаки отказа от использования (неполного использования) системы контроля версий ... 163
Простые системы контроля версий ... 164
Усовершенствованные системы контроля версий ... 164
Для каких файлов нужно использовать контроль версий ... 165

Совместная работа с файлами и клиенты системы контроля версий ... 165
Нет локального клиента, нет общей файловой системы ... 166
Нет локального клиента, но есть общая файловая система ... 166
Есть локальный клиент, но нет общей файловой системы ... 166
Есть и локальный клиент, и общая файловая система ... 166
Проблемы интегрированной среды разработки ... 167
Проблемы графического интерфейса пользователя ... 167
Спецификация SCC ... 168
Интерфейс для веб-браузера или клиент-Java-систем контроля версий ... 168
Основные положения концепции контроля версий ... 169
Советы ... 174
Отслеживание ошибок ... 177
Неконфигурационные средства управления ... 179
ПО для зеркального отображения информации ... 179
Автоматизированное резервное копирование ... 179
Веб-браузер ... 179
Группы новостей в Интернете ... 180
Заключительные комментарии ... 180
Рекомендованная литература, ссылки и ресурсы ... 181

III Математика ... 183

Введение ... 184

Глава 11. Введение в машинные вычисления ... 185
Введение ... 185
Целочисленная арифметика ... 185
Деление и отрицательные числа ... 185
Целые типы и их размер ... 188
Переполнение или исчезновение значащих разрядов ... 189
Математические операции с плавающей запятой ... 192
Неожиданный результат ... 192
Форматы с плавающей запятой ... 193
Погрешности округления ... 195
Ошибки при умножении и делении ... 197
Ошибки при сложении и вычитании ... 198
Обработка ошибок при вычислениях с плавающей запятой ... 200
Использование эквивалентных выражений для устранения катастрофической потери точности ... 202
Арифметические операции с фиксированной запятой ... 204
Область применимости ... 204
Представление чисел с фиксированной запятой и операции над ними ... 205

Обработка ошибок при выполнении операций с фиксированной запятой ... 206
Заключение ... 207
Библиография ... 207

Глава 12. Аппроксимации для вычислений с плавающей запятой ... 208
Общие замечания о тригонометрических функциях ... 209
Косинус и синус ... 210
Более точное вычисление косинуса ... 216
Тангенс ... 217
Более точное вычисление тангенса ... 222
Арктангенс, арксинус и арккосинус ... 223

Глава 13. Математические функции ... 227
Код Грея ... 227
Умножение целого на константу ... 227
Вычисление исключающего ИЛИ ... 227
Извлечение квадратного корня в целых числах ... 228
Важнейшие математические функции ... 228

Глава 14. Стандарт IEEE 754 для чисел с плавающей запятой ... 229
Специальные значения ... 230

IV Системы реального времени ... 232

Введение ... 233

Глава 15. Ядра реального времени ... 234
Введение ... 234
Что такое ядро реального времени? ... 234
Что такое задача? ... 235
Тактовый интервал таймера ... 238
Планирование задач ... 239
Переключение контекстов ... 241
Службы ядра ... 242
Службы ядра. Семафоры ... 242
Службы ядра. Очереди сообщений ... 246
Службы ядра. Управление памятью ... 248
Нужно ли вам ядро? ... 248
Можете ли вы использовать ядро? ... 249
Выбор ядра ... 250
Заключение ... 253

Глава 16. Реентерабельность ... 254
Атомарные переменные ... 254
Еще два правила ... 256
Обеспечение реентерабельности кода ... 257
Рекурсия ... 259
Асинхронность оборудования/микропрограммного обеспечения ... 260
Состояние конкуренции ... 261
Варианты решения проблемы ... 262
Другие ОС реального времени ... 264
Метастабильные состояния ... 265
Микрокод, а не оборудование ... 267

Глава 17. Латентность прерываний ... 271
Получение данных ... 274

Глава 18. Как работает ваш компилятор языка C: минимизация размеров программы ... 277
Современные компиляторы языка C ... 278
Структура компилятора ... 278
Смысл программы ... 280
Базовые преобразования ... 280
Распределение регистров ... 282
Вызовы функций ... 283
Подстановка функций ... 283
Сжатие кода низкого уровня ... 284
Компоновщик ... 284
Управление оптимизацией, осуществляемой компилятором ... 285
Модель памяти ... 286
Советы по программированию ... 286
Правильно подбирайте размер переменных ... 286
Используйте указатели наиболее подходящего типа ... 287
Структуры и байты заполнения ... 288
Используйте прототипы функций ... 289
Используйте параметры ... 290
Не используйте операцию получения адреса ... 290
Не используйте встроенный ассемблер ... 291
Не пишите остроумный код ... 291
Проверяйте значения битовых полей перед использованием ... 293
Следите за использованием библиотечных функций ... 293
Используйте дополнительные подсказки компилятору ... 294
Финальные замечания ... 294
Благодарности ... 295

Глава 19. Оптимизация кода на языках C и C++ ... 296
Устанавливайте размеры структур равными степени двойки ... 296

Размещайте метки «case» как можно ближе друг к другу ... 296
Размещайте наиболее используемые метки case вначале ... 296
Разбивайте крупные операторы switch на вложенные операторы-переключатели ... 297
Минимизируйте число локальных переменных ... 298
Описывайте локальные переменные как можно глубже внутри функций ... 298
Сокращайте число аргументов ... 298
Используйте ссылки при передаче параметров и возвращаемого значения для типов, имеющих длину более 4 байтов ... 299
Не определяйте возвращаемое значение, если оно не используется ... 299
Учитывайте расположение ссылок относительно кода и данных ... 299
Старайтесь использовать тип int вместо char или short ... 300
Пишите облегченные конструкторы ... 301
Старайтесь использовать инициализацию вместо присваивания ... 301
Используйте списки инициализации конструкторов ... 302
Не объявляйте функции виртуальными «на всякий случай» ... 302
Используйте подстановку для функций длиной в 1–3 строки ... 302

Глава 20. Макросы assert в системах реального времени ... 304
Проблемы встраиваемых систем ... 304
Макросы assert в системах реального времени ... 306

V Ошибки и исправления ... 312

Введение ... 313

Глава 21. Реализация загружаемого микрокода с помощью флеш-памяти ... 314
Введение ... 314
Микропрограмматор ... 315
Преимущества микропрограмматоров ... 315
Недостатки микропрограмматоров ... 316
Получение микропрограмматора ... 316
Базовый микропрограмматор ... 317
Типичные проблемы и их решение ... 319
Отладчику «не нравятся» перезаписываемые области памяти ... 319
Отладчикам «не нравится» код, выполняющий перемещение самого себя ... 320
Невозможность генерации позиционно-независимого кода ... 322
Отсутствие микрокода в момент загрузки ... 323
Постоянная блокировка по времени ... 323
Неожиданное отключение питания ... 324
Аппаратные альтернативы ... 325

Разделение кода и данных ... 326
Гибкость и надежность ... 326

Глава 22. Диагностика памяти ... 328
Тестирование ПЗУ ... 328
Тестирование ОЗУ ... 330

Глава 23. Энергонезависимая память ... 337
Контролирующие схемы ... 337
Запись многобайтных значений ... 339
Тестирование ... 343
Выводы ... 344

Глава 24. Профилактическая отладка ... 345
Стеки и кучи ... 345
Заполнение памяти ... 348
Блуждающий код ... 350
Специальные дешифраторы ... 352
Блоки управления памятью ... 353
Выводы ... 354

Глава 25. Обработка исключительных ситуаций на C++ ... 355
Горы (ориентиры безопасности исключительных ситуаций) ... 356
История этой территории ... 357
Коварная ловушка ... 358
Смола! ... 360
Самый легкий путь ... 361
Оператор присваивания – специальный случай ... 363
В плохую погоду ... 364
Подведем итоги ... 367
Литература ... 371

Глава 26. Отличный сторожевой таймер ... 372
Внутренние сторожевые таймеры ... 375
Внешние сторожевые таймеры ... 378
Характеристики отличных сторожевых таймеров ... 379
Использование встроенного сторожевого таймера ... 383
Внешний сторожевой таймер ... 385
Сторожевые таймеры для многозадачной среды ... 387
Выводы и некоторые соображения ... 389

Приложение А. ASCII-коды ... 392

Приложение Б. Выравнивание и порядок байтов ... 394
Ограничения, накладываемые на выравнивание байтов ... 394
Для чего нужно ограничивать выравнивание байтов? ... 394

Общие правила выравнивания байтов ... 396
Выравнивание структур для повышения эффективности ... 396
Порядок байтов ... 397
Почему используется различный порядок байтов? ... 397
Подпрограммы преобразования ... 397

Указатель ... 399

Лицензионное соглашение на использование компакт-диска Elsevier Science ... 403

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 Ганссл,
февраль 2004, 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 (COCOMO) 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, TCP/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.

Содержание CD-ROM

На прилагаемом к книге компакт-диске вы найдете:

- полную электронную версию текста в формате Adobe pdf с поддержкой функций поиска;
- каталог с исходными кодами всех примеров программ, приведенных в этой книге.

Подробности о содержимом CD-ROM – в файле ReadMe.

РАЗДЕЛ СТРАНИЦА

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

щего на веб-сайте 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.

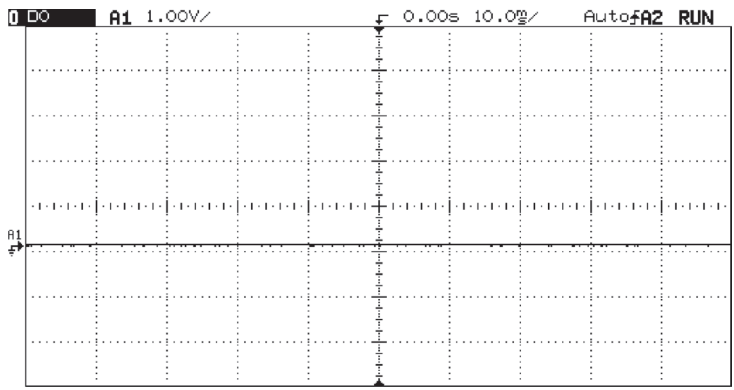

Рис. 1.1. Постоянный ток характеризуется сигналом постоянной, неменяющейся амплитуды

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[1].

Заряды измеряются в кулонах (Кл), 1 кулон равен заряду $6{,}25\times10^{18}$ 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.

---

[1] 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^{-6}	Экономичные логические и аналоговые схемы, ОЗУ с резервным питанием от батарей
Пикоампер	пкА	10^{-12}	Очень чувствительные аналоговые входные цепи
Фемтоампер	фА	10^{-15}	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» (Ω). 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.

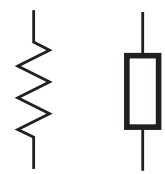

Рис. 1.3. Зигзагообразная линия слева – стандартное обозначение резистора на электрических схемах. Справа приведено обозначение, принятое в Великобритании. Как говаривал Черчилль, «мы – два народа, разделенных общим языком»

Таблица 1.2. Номиналы реальных резисторов

Название	Сокращенное обозначение	Значение [Ом]	Где встречается
Миллиом	мОм (mΩ)	0,001	Сопротивление 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 k)	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)	10^6	Аналоговые цепи с сигналом низкого уровня
Сотни мегаом		10^8 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:

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

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.

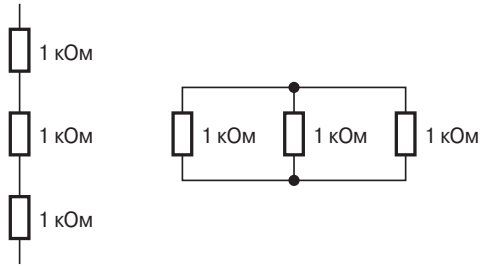

Рис. 1.4. Три последовательно соединенных резистора слева эквивалентны одному компоненту с сопротивлением 3000 Ом. Правые резисторы, соединенные параллельно, работают как одно сопротивление со значением 333 Ом

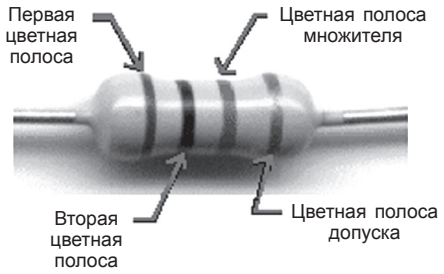

Рис. 1.5. На этой черно-белой фотографии цвет полосок не виден. Однако мы можем прочитать шифр слева направо: две первые описывают целую часть номинала, третья – множитель. Четвертая полоса может быть золотой (5%) или серебряной (10%), характеризуя допуск значения

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»[1]

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

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

Две первые полосы слева определяют целую часть номинала резистора. Третья – это множитель. Прочитайте численное значение, зашифрованное двумя первыми полосами, и умножьте его на коэффициент, определяемый третьей полосой. Например: коричневая-черная-красная = 1 (коричневая) 0 (черная) умножить на 100 (красная), или 1000 Ом, то есть 1 кОм. В табл. 1.4 приведены дополнительные примеры.

Таблица 1.4. Примеры чтения цветовой маркировки и вычисления сопротивления

Первая полоса	Вторая полоса	Третья полоса	Расчет	Значение [Ом]	Общепринятое написание
Коричневая	Красная	Оранжевая	12 × 1000	12 000	12 кОм
Красная	Красная	Красная	22 × 100	2200	2,2 кОм
Оранжевая	Оранжевая	Желтая	33 × 10 000	330 000	330 кОм
Зеленая	Синяя	Красная	56 × 100	5600	5,6 кОм
Зеленая	Синяя	Зеленая	56 × 100 000	5 600 000	5,6 МОм
Красная	Красная	Черная	22 × 1	22	22 Ом
Коричневая	Черная	Золотая	10 ÷ 10	1	1 Ом
Синяя	Серая	Красная	68 × 100	6800	6,8 кОм

Существует набор стандартных номинальных значений выпускаемых резисторов. Новички в схемотехнике часто пытаются заказать компоненты, не существующие на практике; более опытные инженеры знают, что, например, резисторы с сопротивлением 1,9 кОм не выпускаются. Инженерия – весьма прикладное искусство; хороший конструктор всегда использует стандартные и легкодоступные компоненты.

Электрические цепи

Электрический ток всегда течет по замкнутому контуру. Отсоединенная от цепи аккумуляторная батарея разряжается очень медленно, поскольку нет замкнутого контура, нет никакого контакта (за исключением ненулевого сопротивления водяных паров воздуха) между ее выводами. Чтобы лампочка загорелась, подсоедините концы провода к клеммам батареи; теперь электроны смогут двигаться по замкнутому контуру от отрицательного вывода батареи через лампу и снова к батарее.

Есть только два типа электрических цепей: последовательные и параллельные. Все существующие схемы являются их комбинацией. В *последовательной цепи* нагрузки подключаются по кругу друг за другом, ток также течет по кругу, проходя через все нагрузки по очереди. В последовательной цепи сила тока одинакова на всех нагрузках.

Все параметры последовательной цепи рассчитываются просто. На рис. 1.6 12-вольтовая аккумуляторная батарея служит источником питания для двух последовательно соединенных резисторов. Согласно закону Ома сила тока, протекающего в цепи, равна напряжению (в данном случае 12 В), деленному на сопротивление (сумму сопротивлений двух резисторов, или 12 кОм). Следовательно, сила тока: