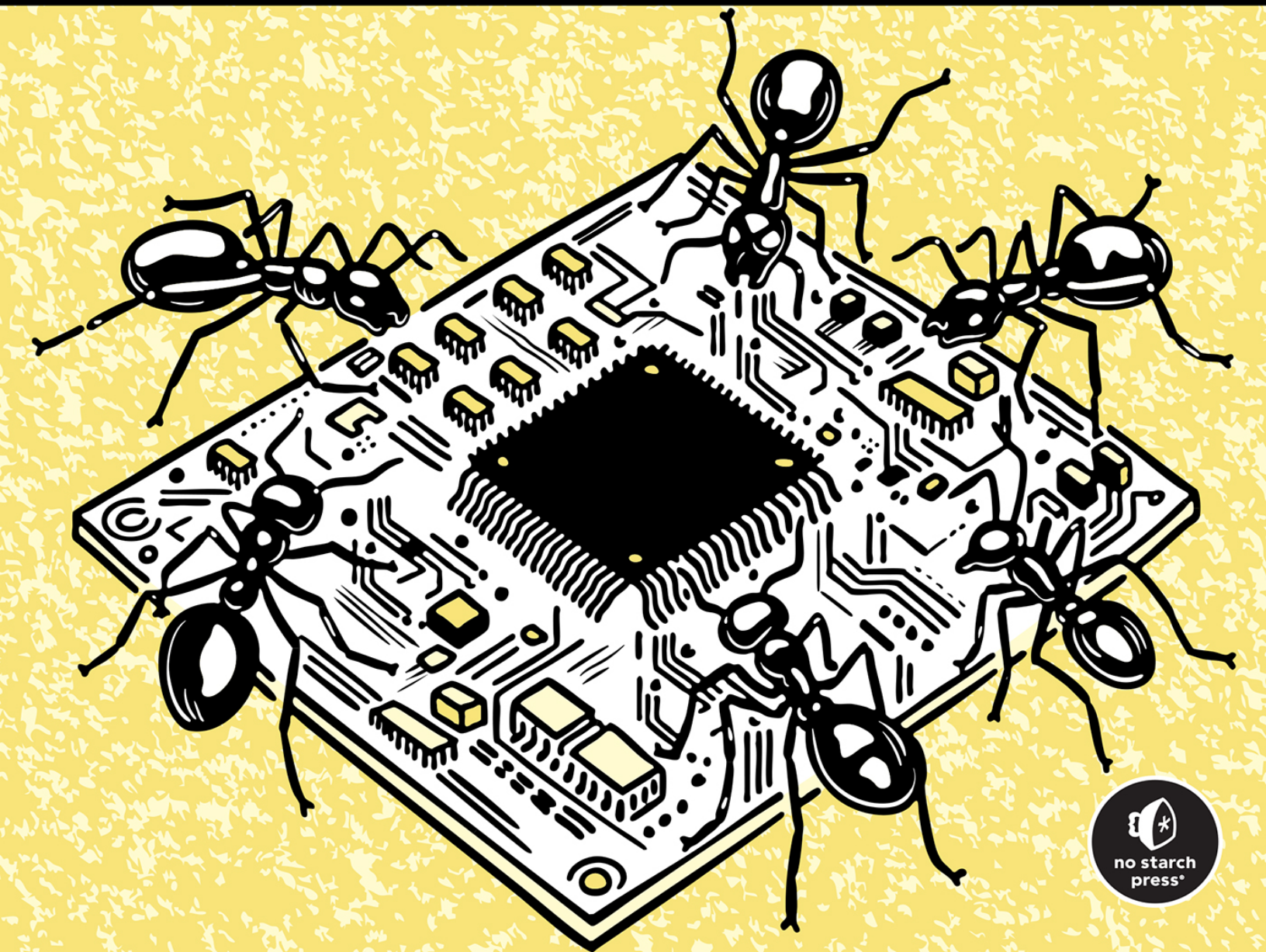
DESIGNING ELECTRONICS THAT WORK

REAL-WORLD HARDWARE DEVELOPMENT

HUNTER SCOTT

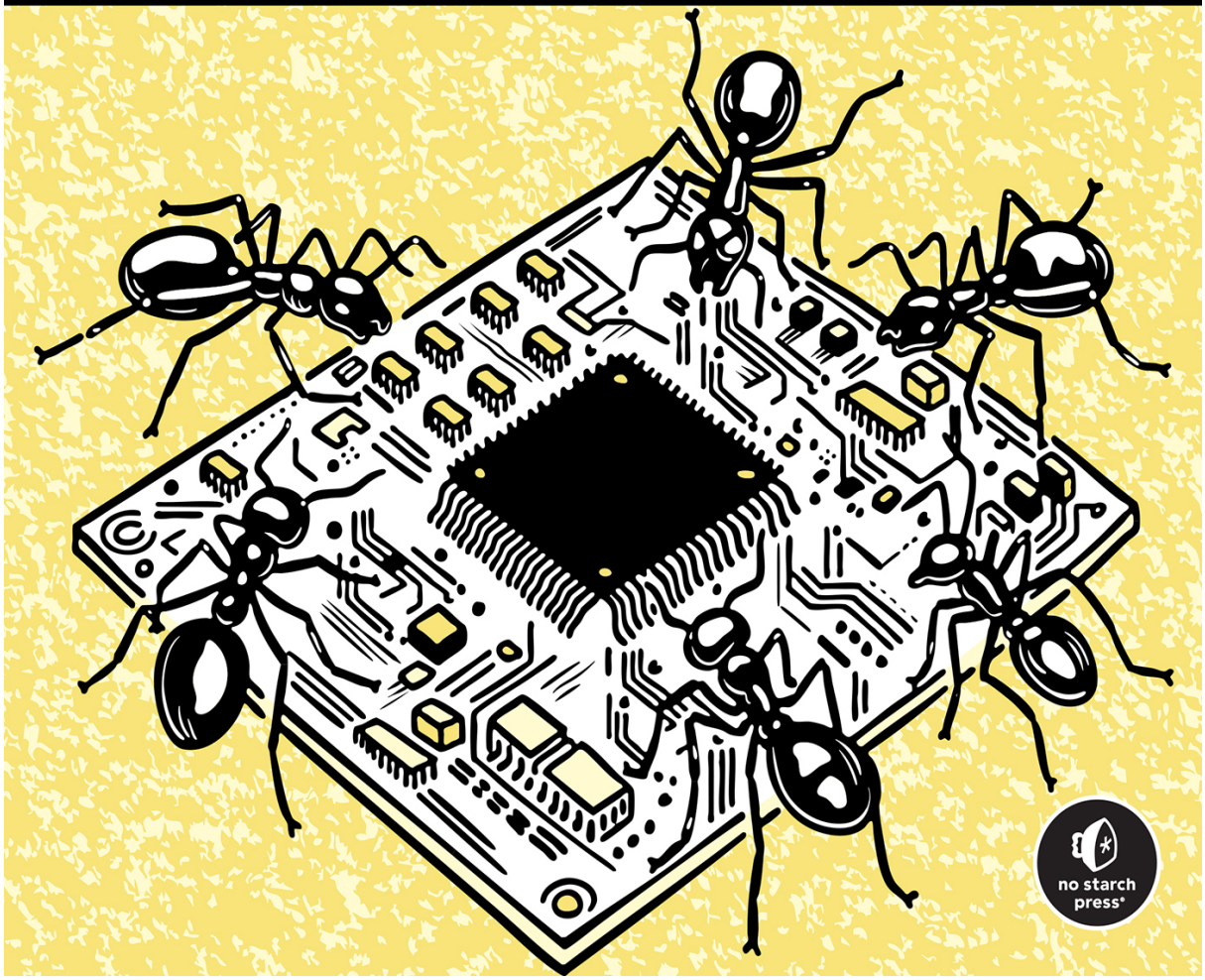


no starch press®

DESIGNING ELECTRONICS THAT WORK

REAL-WORLD HARDWARE DEVELOPMENT

HUNTER SCOTT



no starch press®

DESIGNING ELECTRONICS THAT WORK

Real-World Hardware Development

by Hunter Scott

no starch press®

San Francisco

First printing

29 28 27 26 25 1 2 3 4 5

ISBN-13: 978-1-7185-0336-6 (print)
ISBN-13: 978-1-7185-0337-3 (ebook)

Published by No Starch Press®, Inc.
245 8th Street, San Francisco, CA 94103
phone: +1.415.863.9900
www.nostarch.com; info@nostarch.com

Publisher: William Pollock
Managing Editor: Jill Franklin
Production Manager: Sabrina Plomitallo-González
Production Editor: Sydney Cromwell
Developmental Editor: Nathan Heidelberger
Cover Illustrator: Josh Kemble
Interior Design: Octopod Studios
Technical Reviewer: Eric Schlaepfer
Copyeditor: George Hale
Proofreader: James Brook
Indexer: BIM Creatives, LLC

The following images are reproduced with permission: Figures 7-16 and 7-17 courtesy of NASA Goddard Space Flight Center; Figures 6-18, 12-3, and 12-4 courtesy of Greg Davill.

Library of Congress Control Number: 2025016561

For customer service inquiries, please contact info@nostarch.com. For information on distribution, bulk sales, corporate sales, or translations: sales@nostarch.com. For permission to translate this work: rights@nostarch.com. To report counterfeit copies or piracy: counterfeit@nostarch.com. The authorized representative in the EU for product safety and compliance is EU Compliance Partner, Pärnu mnt. 139b-14, 11317 Tallinn, Estonia, hello@eucompliancepartner.com, +3375690241.

To K and the kids

About the Author

Hunter Scott is an electrical and computer engineer with 15 years of experience in designing and implementing advanced hardware systems. He holds a degree in computer engineering from Georgia Tech and has taken two hardware companies through Y Combinator as a founder. He is a founding engineer at Reach Power, a long-range power beaming startup in Silicon Valley, where he leads technical strategy. He also advises startups and is a venture partner at Pioneer Fund. Scott has worked on a wide range of electronics designs, including medical devices, robotics, communications systems, and high-power millimeter wave phased arrays, and he holds several patents. His personal projects have included designing electronics for a large interactive art installation at Burning Man and developing a Twitter bot that won hundreds of contests. You can learn more about him or get in touch at *http://hscott.net*.

About the Technical Reviewer

Eric Schlaepfer coauthored a book of cross-sections, *Open Circuits* (No Starch Press, 2022), and runs an engineering-related Bluesky account, @*tubetime.bsky.social*, where he posts cross-section photos, shares his retrocomputing and reverse engineering projects, investigates engineering accidents, and even features the occasional vacuum tube or two. Some of his better-known projects include the MOnSter 6502 (the world's largest 6502 microprocessor, made out of individual transistors), the Snark Barker (a retro re-creation of the famous Sound Blaster sound card), and the Three Fives and XL741 transistor-scale replica chip kits. His diploma, a BS in electrical engineering from California Polytechnic State University, San Luis Obispo, was signed by Arnold Schwarzenegger.

BRIEF CONTENTS

CONTENTS IN DETAIL

6
LAYOUT DESIGN

Keys to Success
Designing for Performance
PCB Physics
Design Tips
Testing Considerations
Creepage and Clearance Requirements
High-Speed Design Considerations
Stackup Design
Layers
Substrates
Traces
Vias
A Stackup Example
Thermal Considerations
Common Gotchas
DRC Checks
Footprint Mistakes
Conclusion

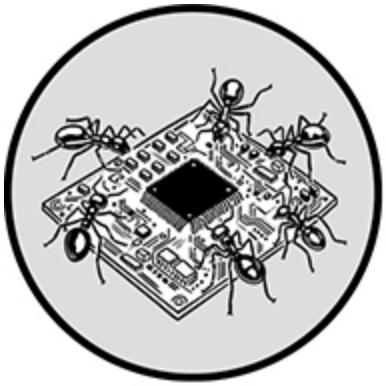
7
DESIGN FOR EXCELLENCE

Design for Fabrication
Fabricator Constraints
PCB Markings
Copper Thieving
Panelization
Design for Assembly
Fiducials
Markings to Aid Assembly
Soldering Considerations

ACKNOWLEDGMENTS

Thank you to my parents and family for the support and opportunities you gave me, and for putting up with the volume of electronic junk I've accumulated. Thank you to everyone who has acted as a mentor to me: students and professors at Georgia Tech, colleagues at Motorola and Reach, and the YC community. Thanks for sharing your own lessons learned, and thank you for being patient with me as I made and learned from all the mistakes that this book covers. Thank you to the team at No Starch Press for making this book vastly better.

INTRODUCTION



Successful electronics design is all about the details. From the tolerances of the components you choose to the precise alloy of solder you use, every detail matters. You'll find yourself accumulating a personal toolkit of techniques and tricks to manage these details as you design and implement products and devices. What might seem like minor details at first—moving a connector to a different side of the board, an extra bypass capacitor, a modified trace width—can cascade into major impacts downstream, with careful application saving a huge amount of time and money.

This book is a collection of the essential and often overlooked details of electronics design. It's full of tips, techniques, and tricks that may take just moments to implement but can have a disproportionate effect on your design's success. The guidelines in the book are immediately applicable, and I learned almost all of them the hard way: by screwing it up the first time. By reading this book, hopefully you can avoid those same screwups.

What This Book Is

This book is a guide to designing and manufacturing electronics, covering the things most people learn onlythrough experience. It contains an abundance of tricks and techniques that save designers time and money by helping them speed up development and avoid mistakes. It's concise, unique, and most of all, useful. It enables anyone with entry-level electrical engineering knowledge to build elegant hardware that works on the first try. Furthermore, it explains the reasoning behind each method so your designs are driven by insight, not by blind rule following.

Most electronics books answer questions like "What is a capacitor and how does it work?" This book instead addresses questions like "How do I know which capacitor to buy out of the hundreds of thousands available that all look the same? And how do I use that capacitor in my schematic and layout so it performs in the way I expect?" These kinds of questions aren't well covered in electronics books or even in school, but if you don't know the answers, your design won't work, and you'll waste days or weeks troubleshooting.

Who This Book Is For

This book is written for new electrical engineering graduates, hobbyists, startup founders, hardware hackers, artists, researchers, grad students, people who want to move past Arduinos and make custom printed circuit boards (PCBs), and anyone who wants to get their designs back from fabrication and have them work on the first revision. It contains time-saving techniques and helps you catch bugs before they're fabricated. It cuts through the fluff and gets straight to what you need to know and how to apply it.

I'm assuming you have at least a basic understanding of electrical engineering. Again, I won't waste time telling you what a capacitor is, but rather I'll focus on how to pick the right one and use it effectively. While the book is oriented toward product design and implementation, it's equally applicable to hobbyist projects, art installations, and other amateur electrical engineering projects. If you're designing your first PCB, reading this book may help you avoid mistakes you didn't know you were going to make. If you've designed hundreds of PCBs, this book will act more as a checklist to help illuminate any area you forgot to think about.

Some parts of this book may seem like they're applicable only to the kind of work a professional electrical engineer would do at a company. While it's true that some of the more rigorous, process-based guidelines we'll discuss aren't relevant to a hobbyist, the high-level concepts behind those guidelines *are* still applicable and useful. For example, it's probably unnecessary to write a traceable requirements document for a side project, but thinking about your design goals beforehand, rather than just adding and removing features willy-nilly, can help you actually finish your projects, prevent feature creep, and keep your costs down. Before you dismiss a section as too "corporate," consider how the spirit of the idea can be helpful to you.

This book was written to be relevant to the design of many different kinds of electronic devices, but it doesn't cover all information for all classes of devices. If you're designing something that requires special

certification, safety, or reliability constraints (like a medical device, spaceflight-rated device, or safety-critical device), use this book only as a reference and defer to the required standards. Perfectly following all of the guidelines in this book won't guarantee that your design will pass all certifications and work on the first try, but it will help you avoid many common and preventable problems.

What's in the Book

This book is laid out to walk you through the process of figuring out what to build; doing the design and prototype work, including picking parts and making a schematic and layout; and fabricating, assembling, and testing the design. It also includes chapters on other useful topics, like what equipment to buy and how to figure out why your circuit isn't working. The book is roughly in chronological order of the full product-engineering process. Here's a more detailed breakdown of what you'll learn in each chapter:

Chapter 1: What to Build and How to Plan for It Covers the basics of conducting product research, talking to users, writing requirements and specifications, and scheduling. The chapter also provides a high-level overview of the full production process.

Chapter 2: Component Specifications and Purchasing Outlines how to make sure your components will meet your specifications and the different methods commonly used to purchase parts.

Chapter 3: Selecting Passive Components Offers specific advice on a wide range of passive components, including resistors, capacitors, inductors, cables, connectors, and circuit protection. How do you know which capacitor you should buy? What parameters should you care about on a connector? This chapter will tell you.

Chapter 4: Selecting Active Components Features guidance on a wide range of active components, including oscillators, ADCs and

DACs, power supplies, microcontrollers, RF components, transistors, diodes, and batteries.

Chapter 5: Schematic Design Shows how to draw a schematic so that it's easy to read and understand. Illustrates the conventions of schematic design and highlights common problems to avoid so your design will work well.

Chapter 6: Layout Design Transitions from an abstract schematic to a physical layout design, with an eye toward the actual physics at play in a PCB. Covers high-speed design, stackup selection, and via types.

Chapter 7: Design for Excellence Discusses key design aspects that ensure your PCB can actually be manufactured, including design for fabrication, design for assembly, design for test, and design for reliability.

Chapter 8: Regulatory Requirements for Electromagnetic Compatibility and Immunity Covers how to get your board to pass FCC or other regulatory certification for electromagnetic interference and compatibility. Even if you're not planning to get certified, this chapter will help you avoid difficult-to-diagnose problems.

Chapter 9: Cost Engineering Demonstrates how to reduce the cost of your product without making it a piece of junk. Looks separately at cost engineering during schematic design, layout design, and assembly.

Chapter 10: Prototyping Discusses how to quickly prove out an idea by hacking stuff together. Considers different prototyping platforms and offers tricks to test an idea fast and on the cheap. Also touches on maintaining a lab notebook.

Chapter 11: Building a Lab Offers guidance on what kind of equipment you need, where to get it, and how to build a lab even with a modest budget. Covers soldering and testing equipment, as well as essential lab safety.

Chapter 12: Fabrication and Assembly Addresses how to find someone to make your PCB and solder the parts to it, and how to do that yourself. Includes tips on how to become a soldering pro and strategies for getting all of your subsystems to integrate together.

Chapter 13: Testing Helps you figure out what tests your product actually needs to pass and walks you through designing those tests to realistically evaluate whether your product will survive contact with the real world. Discusses different regulatory tests, the board bring-up process, and how to design test fixtures.

Chapter 14: Troubleshooting Suggests what to do when things go wrong with your product. Outlines common ways electronic devices fail and lays out different strategies and mental models for working a problem and getting to the real root cause.

Appendix A: How to Give a Demo Touches on demonstrating your product for customers or investors in a compelling way. Shows how to minimize the chance of things going wrong and how to sell the vision for your product, even if it's not production-ready yet.

Appendix B: Recommended Resources Provides an organized list of companies that can help you take your product from an idea to a real device shipped to millions of customers.

Appendix C: Example Fabrication Notes Contains a template you can use for sending fabrication notes to your PCB manufacturer to make sure they understand what you want and you can preemptively answer their questions.

A Note on Mistakes

This whole book is designed to help you avoid mistakes. There are dedicated sections on avoiding mistakes in particular areas like schematic design and testing. However, there's one critical technique for avoiding mistakes that this book can't do for you: You need to have other people review your design. They'll catch things you missed either

because you've been staring at the schematic for a week straight or because they have experience (or have made mistakes!) that you don't.

Get multiple people to look at your design, ideally from different engineering teams (electrical, industrial design, manufacturing, and so on). That said, don't rely on your reviewers to catch all of your mistakes. Most people will look only at the high-level system design, so don't expect them to be particularly perceptive to mistakes below that level. You must still own your design, take responsibility for it, and be intimately familiar with it.

Despite your best efforts, you will inevitably make mistakes. It's easy for novice and experienced designers alike to get frustrated at themselves when they discover they've made a silly mistake. I struggled with this for a long time because it made me think I was a terrible engineer whenever I found a stupid mistake. However, mistakes are totally normal, and you can't let them discourage you.

I've talked to some of the most respected engineers in Silicon Valley, who have designed influential products and have dozens of patents and deep domain expertise. I once asked one of them how he got his designs right on the first try. His response? "Oh, you don't. Plan into the budget and schedule time for at least two PCB spins [revisions] because you'll always have to change something or find out you got a footprint wrong or something." One engineer told me that he specifically remembered the first time he got a design completely right on the first try. That should give you an idea of how rarely it happens, even for professionals.

Spending too much time trying to get everything perfect can actually be bad, especially during design and prototyping. It slows everything down, puts unnecessary pressure on people, and can kill creativity. In these early stages, it's often more productive to try things out, get iterations out the door, and see what works. The good thing about silly mistakes is that they're usually easy to fix. I'll take an easy-to-fix silly mistake over a subtle bug that's hard to find any day.

Work on your design until you're 90 percent sure it will work, and then fabricate it. You'll never go from design to shipping in a single iteration anyway, and even if you do get it perfect, the mechanical engineers will probably want you to change the dimensions of the board

or something. The point is, it's okay to make mistakes. It can be really frustrating when the mistake you make is simple or avoidable, but when that happens, just write it down, figure out a fix, and move on.

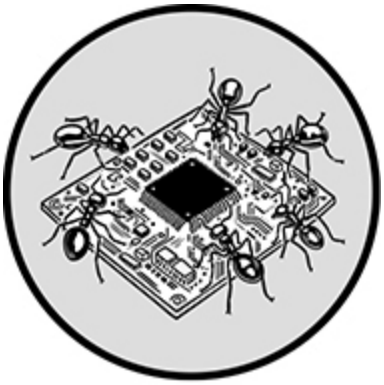
Online Resources

You can get links to the equipment and tools mentioned in this book on the book's website, *https://designingelectronics.com*. There are also links to further reading and software tools.

PART I
PLANNING

In Chapters 1 through 4, you'll learn how to talk to users, define your product, and decide what to make. We'll then discuss how to identify the components you need and how to get them. Finally, we'll do a deep dive into choosing passive components like resistors, capacitors, and inductors, as well as active components like power supplies and microcontrollers. You'll be able to figure out which parts from among the millions of available options are best for your design.

1
WHAT TO BUILD AND HOW TO PLAN FOR IT



In this chapter, we'll explore techniques for coming up with useful product requirements so you can develop a clear picture of what you want to build. Then, we'll see how to translate those requirements into technical specifications. We'll also discuss how to plan for production by predicting an accurate schedule. Finally, we'll look at ways to keep a project on track and review the main steps in the production process.

When you're learning how to be an engineer, whether in school or on your own, you're mostly focused on *how* to build things. However, figuring out *what* to build is even more important and sometimes just as hard. Making progress on a design is fun and satisfying, which can make it easy to build the wrong thing. You can confuse progress on your design with progress on your product. Developing a correct set of product requirements will prevent this by giving you a concrete idea of what you want to build before you start building it.

Requirements and Specifications

Every design is driven by both a set of requirements and a specification. *Requirements* are the things a device must be able to do. In contrast, a product *specification* lists the technical implementation details needed to meet the product requirements. A list of product requirements is typically shorter than a full product specification, but the length of both depends on the complexity of the device.

If you're lucky, you'll receive both product requirements and specifications that are reasonable and achievable. However, more often than not, you'll need to create the requirements, the specification, or both. In this section, we'll explore strategies for defining a project's requirements and specifications. Our focus will be on requirements and specifications pertaining to electronics design, but it's important to keep in mind that electronics design isn't independent of other disciplines, such as mechanical and industrial design. These other areas will inevitably affect and influence a product's electrical design, so be aware that you'll need to consider not just the *electrical* product requirements and specifications but also other requirements and specifications from disciplines that are outside the scope of this book.

Identifying Product Requirements

Product requirements are typically high-level statements about what a product should be able to do. These requirements don't normally include detailed information about the technical implementation; that's what the specification is for. For example, say that you want to develop an underwater camera. A partial list of your product requirements might be:

1. Shoots 1080p video with audio
2. Syncs video with PCs
3. Works up to 30 feet underwater
4. Has battery life of at least one hour

Each item in the list identifies one aspect of what the camera should be able to do, without reference to the materials, components, or other technical details needed to achieve that goal.

Writing a list of requirements is the first thing you should do when starting a new design. That's why this is the first chapter in the book. The decisions you make as you identify the product requirements will drive all of the other decisions you make in picking parts, doing layout, and assembling and testing the device. Few engineers look forward to writing product requirements because they don't get to solder anything or chew on any interesting circuit design problems. However, the requirements drive the entire project. If you get them wrong, it doesn't matter how genius and elegant your design is because no one will ever see it. People won't want to buy the product; or worse, it may never even make it out of the factory.

Writing Effective Requirements

Writing effective product requirements is an art, and it helps to follow some guidelines. Leanna Rierson's book *Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance* has an excellent list of qualities that good software requirements should have. Here's a hardware-adapted version of her list:

Atomic Each requirement should be a single requirement.

Complete Each requirement contains all of the necessary information to define the desired system functionality. The requirement should be able to stand alone without further amplification.

Concise Each requirement should simply and clearly state what must be done and only what must be done. It should be easy to read and understand—even by nontechnical users of the system. In general, a requirement should contain no more than 30 to 50 words.

Consistent Requirements shouldn't contradict or duplicate each other, and they should use the same terminology throughout. Requirements aren't the place to practice creative writing.

Correct Each requirement should be the right requirement for the system being defined and should convey accurate information. This attribute is ensured by the requirement's validation effort.

Implementation-free Each requirement should state what's required without identifying how to implement it. In general, requirements shouldn't specify design or implementation. However, there may be some exceptions, such as specific interface requirements (like USB, MIDI, or Ethernet) or derived system requirements. A *derived system requirement* is a requirement that's implied by another requirement. Returning to our underwater camera example, needing to work 30 feet underwater creates a derived system requirement of an IP68 rating.

Necessary Each requirement should state an essential capability, characteristic, or quality factor. Removing the requirement would create a deficiency.

Traceable Each requirement should be uniquely identified and easily traceable to lower-level requirements, design, and testing.

Unambiguous Each requirement should have only one interpretation.

Verifiable It should be possible to confirm the implementation of each requirement. Therefore, requirements should be quantifiable and include tolerances, as appropriate. Each requirement should be written such that it can be verified by review, analysis, or test. Except for rare cases, if you can't observe the behavior during verification, you should rewrite the requirement. For example, negative requirements are generally not verifiable. Something like "the product shouldn't overheat" is vague, and it's unclear how to test it. Do you just turn the product on and wait? How long is long enough? What does it even mean to over-heat? This requirement would be better rewritten as "the product shall pass all built-in self-tests at an operating temperature of 50°C."

Viable Each requirement should be able to be implemented, usable when implemented, and helpful to the overall system construction.

As mentioned, product requirements shouldn't include negative statements about what a product *shouldn't* do. That said, it can be

valuable to think in negative terms as you prepare to write the product requirements. It can help differentiate your product from that of a competitor or improve the product's usability and experience. Identifying what the product shouldn't do can also help you avoid scope and feature creep, since you'll be explicitly deciding where to draw the boundaries for the product. We'll look closer at feature creep in a moment.

Remembering Overlooked Requirements

Some product requirements will be relatively obvious, like a camera's resolution and battery life, but there are plenty of more subtle requirements that you shouldn't overlook. If you're building a medical device, for example, does it need to survive being sterilized in an autoclave? That's going to change your temperature requirements. Does it need to be biocompatible? Will it be undergoing compression and decompression cycles in an aircraft? The following list contains some (but not all!) commonly missed requirements that may be relevant to your design:

- Operating temperature
- Operating altitude
- Vibration
- Shock/impact resistance
- Physical size
- Finger/hand intrusion
- Solid particle protection (dust, tools, insects, and so on)
- Liquid ingress (dripping water to full submersion)
- Plasticity/flexibility
- Pressure
- Enclosure thermal/electrical conductance
- Security/encryption

- Shelf life for products that contain a battery

Depending on how you formulate them, some of these particular requirements might be better suited for inclusion in the product specifications instead. However you decide to write your product requirements, select them to allow maximum longevity of the product, without over-engineering.

When you're writing your requirements, think about the product's whole life cycle, not just how your customers will use it when they first get it. It's easy to overlook that a product will eventually grow old and become obsolete. With that in mind, here's an exercise that all hardware engineers should do at least once in their lives: Visit the nearest dump. The bigger the better. Get out of your car and look at how big the dump is. Think about the billions of pieces of trash in there. Consider that everything that isn't organic matter was once a manufactured product. Engineers designed it, produced it, and sold it. Maybe people used it for a generation, or maybe they used it for an hour. It doesn't matter anymore because now it's sitting in a huge pile of other products waiting for . . . nothing.

As hardware engineers, we build physical things. That means at some point, everything you design and manufacture is almost certainly going to end up in a dump. That doesn't mean you should think of engineering as a nihilistic exercise in futility. It *does* mean that you should think about the materials you design into your product, the product's reliability, and its repairability. Packaging can be made from recycled materials. In some cases, parts can even be designed for reuse between products. For example, power cables can be made removable and use a standard interface, like those in the IEC 60320 standard.

It's not always possible to design for repairability, but products that are easy to repair can be made to last longer. Remember that it may not just be users who are repairing your product; your technicians may repair devices that have been returned by customers so they can be refurbished and resold.

If a hazardous material must be used in your product, design for recyclability and safe disposal, and make this a product requirement as

well. For example, sealed modules or cartridges containing the hazardous material are easier to recycle or remove and isolate than a reservoir that can spill or requires destructive disassembly. Your design should also be lead-free and RoHS (Restriction of Hazardous Substances) compliant. Fifteen years ago, meeting RoHS requirements could be challenging because a lot of parts still contained lead. Nowadays, it's possible to design a PCB that's lead-free without even realizing it.

You should think about all these things from the get-go when you're writing the product requirements. Make sustainability a requirement so you'll choose materials and processes that are as efficient and ethical as possible. Consider the full life cycle of your product so that one day, when it's sitting in a dump, you'll be able to sleep at night knowing it's not leaching mercury into someone's water supply.

Achieving Requirements and Avoiding Feature Creep

As the number of product requirements increases, achieving total coverage of those requirements gets harder and harder. To demonstrate, Table 1-1 compares the length of the requirements lists for several real software projects with the percentage of requirements actually achieved and the number of lines of code needed to achieve them. Capers Jones collected the data for his book *The Economics of Software Quality* (Addison-Wesley Professional, 2011).

Table 1-1: Requirements Length vs. Completeness

Lines of code	Pages of requirements	Requirements completeness
1,000	14	97%
10,000	115	95%
100,000	750	80%
1,000,000	6,000	60%

Notice that the number of lines of code quickly increases as the requirements grow. At the same time, the more requirements there are, the lower the share of requirements that are actually implemented. One way to maximize the share of requirements you complete is to minimize the number of requirements in the first place. This means breaking large projects into subprojects and, above all, avoiding feature creep.

It's easy for feature creep to first materialize in the product requirements. You begin to say things like, "We already have X, so why not add Y?" These sorts of questions may seem innocuous at first, but before long you could end up with a product that doesn't do any one thing really well, costs a lot of money, and that no one will want to buy. At the other extreme, however, being too minimalist will result in a product that becomes outdated before it even hits the shelf. Don't be afraid to add requirements that are motivated by your market and your customers, but beware of feature creep.

Talking to Users

How do you pick beneficial product requirements without succumbing to feature creep? The short answer is to talk to your users. Find the people that you're building your product for, take them to a coffee shop, and let them talk about what product they want and how they'll use it. You'll need to do this a lot, with a lot of different people. A product may require dozens and dozens of interviews.

The answers you get from people won't be your product requirements quite yet. There's a famous quote from Henry Ford that goes, "If I had asked what people wanted, they would have said a faster horse." The objective of these meetings isn't to build exactly what most people describe. Instead, it's to extract product requirements from what they tell you. The best way to do this is to ask "why" a lot. If someone says they want a faster horse, try to understand what the underlying reason is, and then address that reason in your product requirements. The way you'll actually achieve those requirements will be in the product specification, which comes later.

Talking to users to get useful product requirements is harder than most people think. I highly recommend the book *The Mom Test* by Rob

Fitzpatrick (2013) as a guide to doing this correctly. It's the best book I've ever read on the subject. The main idea is that if you ask your mom if your product idea is good, she'll lie to you because she loves you. In fact, most people will lie to you a little bit if you ask them directly about your product, so *The Mom Test* describes ways to ask questions about the *need* for your product rather than the product itself. This helps you figure out if you're really solving a problem, or if people just think it's a good idea but wouldn't actually buy or use your product. It's a very short book, so pick up a copy.

Some people talk about "defining your users" when writing product requirements. However, you don't *define* your users; you *discover* them. They may not be the people you thought they were. They may not even exist. The only way to find out is by talking to the people you think will be your users and following the trail until you get to the elusive product-market fit. Don't worry so much about what your competitors are doing, either. The most important thing is to make something people want.

Changing Requirements

Product requirements can be allowed to change over time. However, this gets more and more expensive—in both time and money—as the project progresses. If you need to make any changes to the requirements, aim to make them as early in the process as possible.

An easier proposition than changing requirements is to drop some requirements. The feedback loop for which requirements you use and which requirements you drop should again be driven by talking to users. It's extremely important to build beta hardware and get it in the hands of your users so you can discover the changes you need to make to your requirements. Doing so will check assumptions you made about how the product will be used and how intuitive it is to use.

To really get honest feedback, you need to make your beta users pay you money for your device. Money raises the stakes. If someone has to give you their own money for a product, they'll be sure to let you know if they think it was worth it or if they're not satisfied. In the beta stage, you may feel weird about making someone pay for an Arduino stuffed

inside a crude, 3D-printed enclosure, and indeed, some people will refuse to buy it. But if you can find people who will pay you for even a crappy solution to their problem, you not only will get honest feedback from your first users, you'll also have shown that the problem you're trying to solve is bad enough that people are willing to try even an ugly solution.

Writing Specifications

Once you've identified your product requirements, the next step is to write the product specification listing the technical details that will allow you to achieve the product requirements. Returning to our underwater camera example, here's a possible list of product specifications corresponding to the requirements we identified on page 4:

- Contains 1920×1080 RGB CMOS sensor capable of 60 FPS [1]
- Capable of capturing 24-bit audio [1]
- USB-C [2]
- Enumerates as USB Mass Storage device [2]
- MicroSD card slot with support of SDHC Class 6 [2]
- File transfer over 802.11a/b/g/n [2]
- IPX8 compliant [3]
- Operates from 0°C to 40°C [3]
- Battery charging over USB [4]
- Replaceable battery [4]
- Battery life of one hour with camera on [4]

The bracketed numbers after each specification cite the product requirement that the specification fulfills. Notice that our 4 product requirements generated 11 specifications. There will almost always be more specifications than product requirements because a high-level

product requirement will need several specifications to be fully implemented.

Good engineering starts with good product specification design. Because a well-written product requirement isn't specific to any particular implementation, there will likely be many unique product specifications that can fulfill a given set of requirements. Identifying the right specifications for the project therefore requires equal parts creativity and pragmatism. Ultimately, you'll use the product specification to help select components, calculate the link and power budgets, and estimate the time and money required for the project. As you write the product specification, choose technologies and approaches that will keep the final product robust, low-cost, and elegant.

In an ideal world, no project would ever go over budget or off schedule because the time and money required would be conservatively estimated from the specifications. If any of the specifications required too much time or money, you would change the product requirements to put the specifications back below the threshold. This feedback loop is extremely important in making a product successful.

Keep in mind that you should design your specifications to meet, but not exceed, the product requirements. There's no reason to design something "better" than the requirements if the requirements were written correctly. Designing above what's required only introduces liability and risk. If you feel that it's necessary to design past the requirements, change the requirements.

Staying on Schedule

The best manager I ever had was great for a lot of reasons, but one of the skills I admired most was his ability to accurately plan project schedules. Many engineers have a particularly difficult time predicting how long something will take because they are, by nature, optimists. They tend not to think in terms of how long something will *actually* take, but in terms of how long they would *like* it to take, or how long it would take if nothing went wrong. Other engineers have the opposite

problem: They grossly *overestimate* how long tasks will take in an attempt to buy themselves enough buffer time to account for unknowns.

The process of engineering is full of hidden traps and delays, especially if you're working on something that's never been built before. You'll encounter problems of different difficulty levels, and it's hard to predict how many problems you'll hit and how quickly you'll be able to solve them. To borrow an analogy from Paul Graham's 2009 essay "Relentlessly Resourceful," available at *https://www.paulgraham.com*, you don't know whether you are about to plow through a block of foam or granite. A full discussion of schedule planning and prediction would be a whole other book. Instead, what follows are some strategies for saving time and avoiding delays to help ensure that your production schedule stays on track.

Paying for Speed

One way to speed up hardware iteration time is to pay more money for faster turnaround times from a manufacturer and for faster shipping. The extra cost is almost always worth it. Think about how much paying your engineers costs each day, and compare that to how much it costs to shave a day or two off the PCB manufacturing process or to have parts shipped to you overnight. The time saved can also mean more time to troubleshoot when an unexpected issue arises.

Similarly, rapid prototyping tools like a circuit mill are often good investments if you can afford them. They can add more speed to your iterations by allowing you to test ideas in the same day instead of a week later. You'll still need to have parts on hand (which is why it's useful to keep a reasonable parts library in stock), and you still won't have the same design flexibility that fabricating a PCB in a factory would give you, but rapid prototyping is great for de-risking a design. A circuit mill can be especially useful for antenna development, since no components are needed besides maybe a connector and some matching capacitors and inductors.

Reducing Risk

Risk reduction is critical to keeping a project schedule on track. You don't always have time to start over or begin a new approach if your first plan doesn't work, so the more measures you can take to de-risk a design, the better. In addition to using rapid prototyping, consider building multiple versions of your design in parallel or building contingency plans into the same PCB. Add footprints for parts that you can use if something doesn't work, add subcircuits that you can rely on if you need to, and give yourself plenty of opportunities for easy rework in case it's necessary.

The ability to perform skilled rework (either with a technician or by yourself) can save you entire board spins and remove what would otherwise be at least a week of delay from your schedule. You can't always make the changes you need with just a rework, but you may be surprised to see what kinds of advanced rework are possible. For example, adding or removing discrete parts, adding or removing ICs, and changing trace routing (even on internal layers) are all usually achievable. In some cases, you can't perform a rework that will have 100 percent of the performance or functionality you would get with a new PCB, but it can usually be good enough to de-risk the fix so that you can be confident your new PCB will work before you've fabricated it.

Another way to de-risk a design is to prove out individual pieces of it before you put them all together. Use development boards or use a quick-turn PCB fabricator to implement individual sections of your schematic to verify that everything will work the way you think before you spend all the time and money on the completely integrated system. The larger and more complex the design, the more important this is. Having small prototypes or breakout boards of individual chips before you build the main board can also help keep the software side of the project on schedule, since they can begin programming and developing on ICs much earlier.

Anticipating Delays

Many projects, and even entire companies, have failed because of an unexpected production delay or a delay in getting a component. Source your components as early as possible, and always have a backup supplier.

Identify your critical long-lead components early in the design process and begin procuring them. If you're using any uncommon PCB materials, make sure your fabricator gets them ordered and stocked in time to prevent any delays in fabricating your board. Don't forget to take into account any necessary product development delays, like certification or testing. During assembly itself, remember to take into account steps that take longer, like cure times for adhesives or composites. Otherwise, your production may be on time, but your *rate* of production will be slower than expected, which can still result in delayed shipping.

There's an important date to know every year to make sure your manufacturing doesn't get unexpectedly delayed: Chinese New Year. If you do any manufacturing in China, or if any of the parts or subsystems you need are made in China, or if you do any contracting in China, you'll be affected by the Chinese New Year. Factories generally shut down for about two weeks in February, and it can take up to two weeks after that for factories to get back up to normal production volume. Part of the reason this ramp-up takes so long is that many workers use this break as an opportunity to find a new job, and factories can see a roughly 25 percent reduction in their workforce. That means they'll need to hire and train new workers. Many crowdfunded hardware projects have been delayed because they didn't know about or correctly plan for Chinese New Year. The exact date changes every year, and not everyone takes exactly the same amount of time off, so talk to your contractors and factories to learn exactly when they'll be unavailable and make sure you build that time into your schedule. If you need your product for the first quarter, for example, make sure you place the order so it's completed by the end of January.

Keep in mind that China isn't the only country that celebrates a holiday like this. Vietnam, Thailand, and other Southeast Asian countries also celebrate Lunar New Year to varying degrees. Of course, you can avoid Chinese or Lunar New Year completely if you manufacture somewhere that doesn't celebrate it, like the United States, Mexico, or Europe.

Another big holiday in China is the National Day holiday week during the first week in October. Again, talk to your factory to understand exactly how this will impact your production schedule.

Basic Manufacturing Process Steps

One of the leading reasons a project can fall behind is an inaccurate or incomplete understanding of the manufacturing process. This is especially a problem for crowdfunded hardware products, which are notorious for being almost universally late. If you've never gone through the whole production process for a significant quantity of a new product, it can be easy to underestimate the time it will take. Every product will be a little different, but the product development life cycle usually boils down to the same basic steps, which are summarized in the flowchart in Figure 1-1.

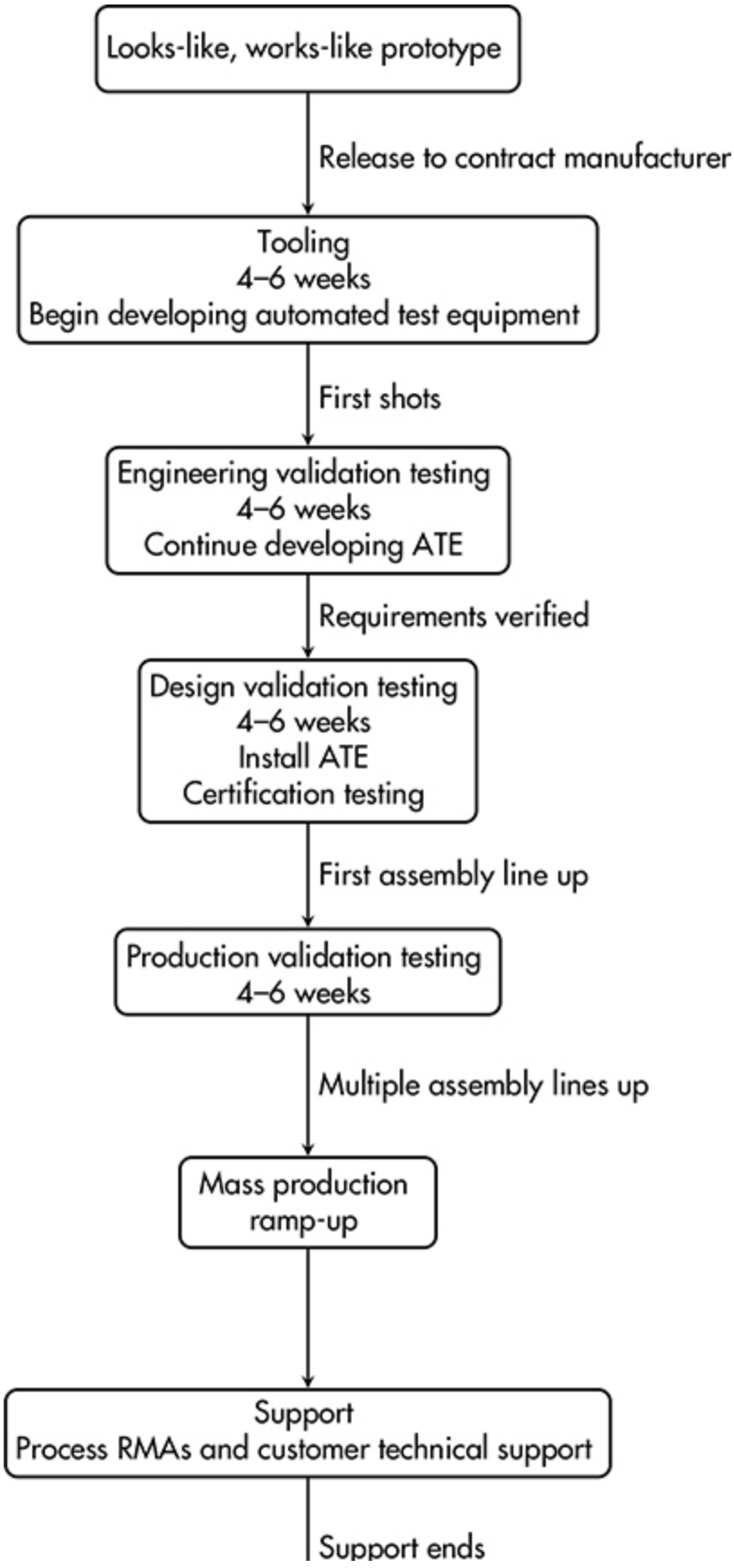
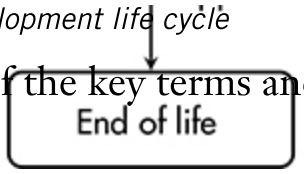

Looks-like, works-like prototype
Release to contract manufacturer
Tooling
4–6 weeks
Begin developing automated test equipment
First shots
Engineering validation testing
4–6 weeks
Continue developing ATE
Requirements verified
Design validation testing
4–6 weeks
Install ATE
Certification testing
First assembly line up
Production validation testing
4–6 weeks
Multiple assembly lines up
Mass production
ramp-up
Support
Process RMAs and customer technical support
Support ends

Figure 1-1: The product development life cycle



Here's a summary of the key terms and stages mentioned in the flowchart:

Looks-like, works-like prototype This is a preliminary version of the product that you build yourself, usually with a 3D-printed enclosure and electronics that are pretty close to what you think the final version will be. Once you're happy with the prototype, you'll select a *contract manufacturer (CM)*. The CM will actually build your design and will work closely with you to help with production and engineering reviews. To start the process, you'll send your design files to the CM's engineering team so they can begin tooling.

Tooling This is the process of creating tools for performing injection molds to produce enclosures or other parts. Tooling usually takes at least three iterations. Tools are often complex devices with moving parts, and they're made of very hard materials that take a long time to machine. Your CM will give you a good estimate of how long this process will take, but be aware that it can take longer depending on how experienced your mechanical engineers are, how good the engineers at the factory are, and how complex your product is. The first test injection-molded parts that come out of your molding tool are called *first shots*. Once these are ready, you'll be able to assemble *engineering models*, the first versions of your product made with injection-molded parts. You'll use these models to begin testing the product.

Automated test equipment (ATE) This includes equipment like a bed-of-nails test jig, programming stations, and test fixtures that quickly and automatically exercise all of your product's functions to verify everything works before it goes out the door. You should begin designing the ATE as early as you reasonably can, but no later than during the tooling process, so it will be ready for use in later phases. This is a big job that's often underestimated. We'll discuss ATE in detail in Chapter 13.

Engineering validation testing (EVT) This phase is where you validate your engineering work by testing your hardware against your requirements. You'll likely need to iterate the PCB at least once during this phase. Move to the next phase only after your requirements have been verified (or mostly verified). Ideally, you'll verify 100 percent coverage of the product requirements during EVT, but requirement verification may be spread out into the next testing phase as well. That said, the more problems you can discover in EVT, the cheaper it will be to fix them.

Design validation testing (DVT) During DVT, you'll verify that your design passes any compliance or regulatory testing, including FCC (Federal Communications Commission, aka wireless certification in the US), CE (Conformité Européenne, a European health and safety certification), RoHS (Restriction of Hazardous Substances, a European environmental certification), and UL (Underwriters Laboratories) testing. More details about DVT can be found in Chapter 13. The units assembled during DVT will have molded parts that were shot on final hard tools. ATE should be completed and installed at this stage, and you should be able to produce units by hand with all final parts.

Production validation testing (PVT) At this point, your first assembly line has been set up and you're debugging mistakes in assembly using the ATE. Units being produced are good enough to be sold to customers. PVT should quickly transition to *mass production (MP)*.

Mass production Multiple assembly lines are brought up, and your main focuses are keeping the yield up, keeping the lines running, and managing suppliers who may be bottlenecking your production.

Support Concurrent to mass production, since customers are now buying and using your product, you'll begin supporting them. Especially early on, this will feed back into your production line to fix problems like premature failures or other unforeseen usability

issues. Your ability to make significant changes will be minimal at this stage unless you can spend a lot of time and money.

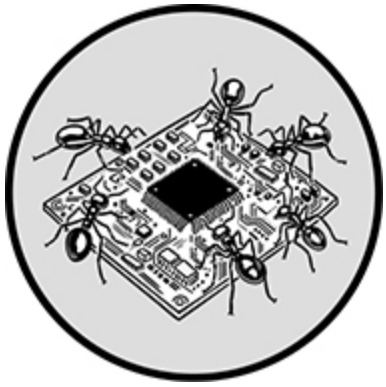
Eventually, after production has ceased and the decision has been made to no longer provide support, the product will be considered *end of life (EOL)*. Planning when EOL will happen is an important part of scheduling. Consider the schedule of your other product lines and how the economics of offering and supporting multiple products simultaneously will impact your business.

Conclusion

In this chapter, we explored strategies for identifying product requirements and translating them into product specifications. Requirements and specifications work together and inform one another, and they play an enormous role in meeting the two most important goals when embarking on a new project: making sure that people actually want what you're building and ensuring that the project will be completed on time and on budget.

We also examined the steps in the production process and looked at ways to keep your project on schedule. Understanding the rhythm of product development will help you collaborate with all of the different people and organizations you need to get your product into the hands of users. A strong grasp of these concepts will make the difference between a painful project and a smooth one.

2
COMPONENT SPECIFICATIONS AND PURCHASING



The art and science of selecting electronic components is rarely taught in school, but it's crucial to the success of a project. The components you choose for a design must meet your specifications and keep your costs low. They also need to be available. Unfortunately, none of these benchmarks are easy to achieve. In this chapter, we'll explore general strategies for determining the specifications that the parts in your design will need and discuss how the purchasing process works.

How to Choose the Right Parts for Your Design

The first goal when picking parts for your design is to ensure that each one meets *all* of your specifications. For example, maybe you find a great LCD screen that has the resolution and refresh rate you need, but will it function under the required environmental conditions? In addition to component characteristics, good part selection takes into account a myriad of other factors, including complexity, robustness, and logistics.

Component Properties

Every component's datasheet is loaded with information about that component, including its tolerance, temperature and voltage ratings, and so on. It's important to understand how these various properties relate to your product specifications. Here are some tips to help you evaluate the suitability of components for your project:

Pay attention to your tolerances. If you're picking out a decoupling capacitor, you probably care a lot less about its accuracy than if you're picking out, say, a capacitor for a high-order filter. Make sure you've gone through your design and specified the required tolerance so the right part gets ordered. If you pick a tolerance that's too tight, you'll spend more money than you need to. However, if you pick a tolerance that's too loose, your circuit might be unreliable or function incorrectly.

Give yourself wiggle room with ratings. Try to choose components and design your circuits so that parts won't be running at more than 90 percent of their specified voltage, temperature, and other ratings. This will help ensure performance, since the manufacturer likely hasn't tested anything above the ratings in the datasheet. If you need to run a part above its nominal conditions or in a nonstandard way, be sure to talk to an application engineer about it, and collect your own thorough test data before you trust the component to operate normally.

Know the failure mode of each part you select. The failure mode depends both on the component itself and on how you'll use it. Will it fail open or closed? Will it get hot? Will it outgas anything? Don't choose a component with an unacceptable failure mode. This becomes especially important with capacitors, and we will cover this in more detail in Chapter 3. For components with an acceptable failure mode, use what you know about how they can and will fail to mitigate that failure. This might drive you to place the component in a particular area of your PCB or choose a certain enclosure design, for example.

Consider automotive-rated components for high temperatures. Many components are manufactured in both a standard operating temperature range and an extended operating temperature range meant for automotive applications. The latter are tested extensively according to stringent automotive standards and are a good way of getting much more robust parts for not much more money. The level above automotive is *mil-spec* (suitable for military use), but parts rated for mil-spec are often significantly more expensive.

Be cautious about inherited hardware or components. Maybe you have some leftover reels of parts from a similar project, or another engineer tells you about a part that has worked well before, or there's an entire module you want to reuse. The danger here is that the requirements may not be exactly the same between projects. For example, if the operating temperature requirements have changed, a component or module that worked great before may now fail. There's certainly nothing wrong with using inherited hardware, and in fact it can save a lot of time and money, but don't assume it will still work. Take the time to ensure it meets all of your new requirements before you integrate it.

If you're ever unsure of anything, keep in mind that every component manufacturer has resources available to help you use their parts successfully, including sales engineers and field application engineers. These people are a free way to parallelize your work. They'll usually assist in part selection and give you a few suggestions of parts that meet your requirements. They know their product lines well, and they want to help make sure your design works—it's literally their job! Take advantage of this, and call them up if you have any questions about functionality, implementation, or performance, or if you're planning on using a part in a nonstandard way.

While most components will have easily accessible datasheets and support from the manufacturer, it can be difficult to locate information about parts that aren't readily available. In this case, try searching the forums at *https://www.elecfans.com*, a Chinese-language website where

users often post hard-to-find PDFs and schematics. You need to register an account before downloading anything from the forum. This can require some trial and error if you don't read Chinese, but Google Translate can help you figure it out.

Component Packaging

The component package is another important consideration when choosing parts for your project, since packaging influences factors like the size, layout, and assembly of a design. Passive components like resistors, capacitors, and inductors most commonly come in package sizes called 2512, 1206, 0805, 0603, 0402, 0201, and 01005. These numbers refer to the part's physical dimensions in hundredths of an inch. For example, 0805 means the part is 0.08 inches by 0.05 inches. However, passive components sometimes follow this naming convention using metric rather than imperial units. This can get really confusing because several metric packages have the same names as imperial packages but are vastly different in size. The four you need to look out for are metric 0402 (which is imperial 01005), metric 0603 (which is imperial 0201), metric 0805 (which is imperial 0302), and metric 1005 (which is imperial 0402). If you aren't aware of this potential confusion, you might accidentally order the wrong size part.

NOTE

Almost everyone in the industry uses the imperial naming convention. This book also uses that convention, so when you see a package size, it's in inches.

For most passive components, 0603 is a good balance between size and ease of assembly. They're large enough to be easily hand soldered but small enough not to take up a lot of real estate on your PCB. Once you use packages smaller than 0402 (like 0201 and 01005), it's difficult to manually apply solder paste with a stencil because the window that the solder paste must squeeze through is so small. Those sizes also all but require the use of a microscope to solder by hand. Even professional

assembly companies often have difficulty working with parts smaller than 0402, which can lead to delays and low yield when they're making your product. Unless space is really at a premium, don't go smaller than 0402.

IC packages like ball grid array (BGA) that pack a lot of pins into a small area require special consideration. They're great for size-sensitive designs or if you need pins with extremely low stray inductance, but they're difficult to rework and require an X-ray to verify correct soldering during production. If size isn't a constraint and an alternative is available, it's best to avoid BGA, CSP, and similar packages.

If you *need* to use a BGA package, consider using an underfill material if the product will be subjected to lots of vibration, shock, or moisture. *Underfill* is an epoxy material that's flooded underneath BGA components and then cured, resulting in a more solid mechanical anchoring to the PCB substrate than just solder balls. Loctite UF 3810 is a good choice because it's strong yet fairly easy to remove with heat if you need to do any rework. That said, using underfill adds complexity to assembly and repair or rework, so you should only use it if you really need it.

The smaller the *pitch* (the distance between two adjacent pins) of a BGA part, the harder it is to route out the signals from the middle of the part. This process is called *escape routing* or *fan-out*. Very dense BGA chips require a high-density interconnect (HDI) PCB to accomplish the escape routing. You'll need blind and buried vias, vias in pad, a large number of layers, extremely narrow traces, or other specialized techniques to get the signals out from under the chip. These processes are expensive, which is another reason why you should avoid BGA if you can. There are companies that have wasted millions of dollars because they committed to a BGA component without realizing the added cost to the PCB until it was too late.

There's one more caution to take when using BGA components: Not all BGA packages are created equal. Especially in BGA components with a large number of pins, the location of power and ground pins on the package has a significant effect on signal integrity and radiated emissions. If power and ground pins are far apart, if there aren't enough

of them, or if they're not evenly distributed, the problems could be bad enough that you need to redesign your product with a different chip. This may seem incredible, and a poorly designed package may not cause problems for every application, but companies have lost a lot of money and even gone out of business because of this issue. An excellent resource for further reading on this is a 2005 paper called "BGA Crosstalk" by Howard Johnson (available at *https://sigcon.com*), which measures the performance of two very similar FPGAs made by two different companies to show that the one with an inferior package design exhibits crosstalk that's about 4.5 times worse.

If you're going to be hand-assembling your design, you can make life simpler by choosing IC packages with exposed pins. This will make it easy to solder with an iron, and any solder shorts will be obvious. The downside is that exposed pins take up more space. ICs with pins underneath the chip take up less space, but they often require hot-air soldering. If you've never used a hot-air soldering iron, it's really not that difficult and is a totally viable method of hand assembly with a little practice.

There are some IC packages, such as quad-flat no-leads (QFN), that have pads underneath the chip but can be soldered without hot air. Instead, the pads in the footprint can be elongated away from the chip and soldered with an iron. Some QFN parts have what's called a *side-wettable flank*, which allows this technique to be used with automated assembly. Automotive manufacturers like this type of package because it allows them to avoid using X-ray inspection.

How to Buy Parts

Actually procuring the parts you've selected is an often overlooked but critical task. Among the many considerations that go into this process, you must decide where you'll order the parts from, how many of each part you need, and whether you'll be able to get them.

Selecting a Distributor

Component manufacturers usually don't sell directly to most customers. Instead, they sell to an electronics distributor. The most popular distributors are DigiKey and Mouser. Other reputable distributors include Avnet, Arrow, and Newark/Farnell. These aren't the only reputable distributors, but they're the ones that most electrical engineers are familiar with.

Not all distributors charge the same amount for the same parts, so it's worth it to shop around for components if you're building a large quantity of anything. There are several online tools that can help you do this. For example, *https://octopart.com* is a free website that allows you to import your bill of materials (BOM) and find the best price for each part from a large list of distributors. It's also great for cost engineering (see Chapter 9 for more information on this). OEMsTrade (*https://oemstrade.com*) is an alternative to Octopart that offers a similar service. Another good resource is *https://findchips.com*, which caters to a more professional market. Unlike Octopart, Findchips isn't free, but it will give you more detailed and historical information about part availability and life cycle to try to help you determine the supply chain risk of using a particular part. SiliconExpert (*https://www.siliconexpert.com*) is a competitor to Findchips that also charges a monthly fee.

Some component manufacturers use more specialized distributors instead of DigiKey or Mouser. The company RFMW is an example of this—they specialize in RF and microwave components. If you go to *https://www.rfmw.com*, you'll see a list of the companies they represent and sell for. This list is also referred to as a *line card*. Manufacturers that use these distributors don't want to spend the time or money on a large sales force, so they contract it all out to a specialized distributor.

Specialized distributors can be really helpful because their sales engineers will assist you with component selection and cost engineering. If you're planning on manufacturing a large number of your product, these sales engineers will also give you free development boards and can connect you to application engineers or other manufacturer tech support that you can't get any other way. They'll also typically give you

the best pricing on a part, even if that part is also available on Mouser or DigiKey.

If you're in China, especially Shenzhen, there are a handful of Chinese-language websites that you can use to find parts and suppliers, including local suppliers. If you can't read Chinese, you'll need to use Google Translate to navigate these websites. Here are some of the most popular ones:

- *https://www.icgoo.net*
- *https://www.ickey.cn*
- *https://www.szlcsc.com*
- *https://www.lcsc.com*

Always try to buy parts from an authorized distributor or the manufacturer directly. This keeps you from receiving counterfeit parts or parts that are damaged due to poor handling or incorrect storage. Before you buy from that questionable overseas distributor that isn't authorized by the manufacturer, ask yourself, "Is saving an extra *n* cents per part really worth the possibility of getting bum parts?" Even worse than all of the parts failing, you may see intermittent failures if only *some* of the parts work. This can send you on a wild goose chase trying to debug your design or assembly process. Or the bad parts could fail once they're in the field, meaning higher-than-expected returns and unhappy customers.

Buying parts from strange sellers gets more tempting when all of your normal distributors are out of stock and you really need parts fast. You can risk it, but be aware that you're definitely taking a risk and that you'll need to perform extra quality assurance on those units. Better to avoid the problem in the first place by making sure all the parts you pick are in stock with at least one distributor, or preferably two. Other companies will sometimes buy out all of the stock of a particular component from one or more distributors for a small manufacturing run. This can result in the unpleasant surprise of seeing a distributor have 2,000 parts in stock one day and then 0 in stock the next.

Stocking issues are also why it's critical to avoid parts that are marked *end of life* and choose only parts that are for active design use. When a part is designated end of life, it's no longer being manufactured or supported. A related term you'll see is *not recommended for new designs (NRND)*, meaning the parts are usually still in production but will soon become end-of-life obsolete. Some distributor websites let you filter parts by their life cycle status, and you should use that feature if it's available. You can also check the website of the original manufacturer or just talk to one of their application or sales engineers. No parts should be marked *obsolete*, and all parts should be marked *active* all the way through the end of your expected production schedule. Manufacturers don't always put this information in the datasheet or on their website, so contact them to confirm if you need to.

Understanding How Parts Are Distributed

When surface-mount (SMT) components are made at their original factory, they're packaged into a long strip of tape. This tape is either a thick paper or a thin plastic with small grooves cut into it for SMT components to sit in, as shown in Figure 2-1. To keep them from falling out, manufacturers stick a thin, clear piece of plastic on top of them.

Figure 2-1: A piece of SMT cut tape

Manufacturers use a similar technique for through-hole components, but the tape holds the leads in place, and the body of the component sticks out sideways. Figure 2-2 shows an example.

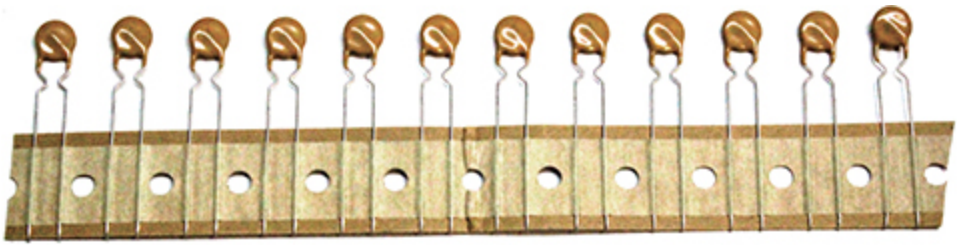


Figure 2-2: A piece of through-hole cut tape

Whether the strip of tape holds SMT components or through-hole components, it will be rolled up into a reel, like the one shown in Figure 2-3, for distribution.

Figure 2-3: A reel of SMT components

If you're going to be populating parts with a pick-and-place machine, you'll need them on a reel. The number of components in a reel depends on how big the component is, but typical numbers are 3,000 or 10,000. Some component distributors like DigiKey will put custom numbers of parts on a reel for you so you don't need to buy 10,000 parts if your build requires less than that. See Chapter 12 for more information on how reels are used.

If you need very small numbers of parts or are assembling by hand, you should order *cut tape* instead of a reel. Cut tape is exactly what it sounds like: small pieces of tape cut from a reel with however many parts you need. Figures 2-1 and 2-2 showed examples. Many

distributors like DigiKey and Mouser will let you order even single parts as cut tape. If you have loose cut tape that you need to use on a pick-and-place machine, there are companies that will re-reel parts for you. Some assembly companies will also accept cut tape and re-reel it themselves.

Components almost always come in special packaging to protect them. The main dangers to components before they get to you are physical damage, moisture, and electrostatic discharge (ESD). To protect against ESD, parts come in special sealed bags that have a static-dissipating coating. To maintain ESD safety, parts should be removed from these bags only at a static-safe workstation. To protect against physical damage, parts sometimes come nestled in antistatic foam or bubble wrap. Moisture is dangerous because it can migrate into components and cause them to fail during reflow when the trapped moisture is heated and expands. Different parts have different levels of moisture sensitivity, so a rating system called the moisture sensitivity level (MSL) was created. Table 2-1 outlines this system (the full standard is in JEDEC J-STD-033B.1).

Table 2-1: Moisture Sensitivity Level

Level	Floor life
1	Unlimited at <= 30°C and 85 percent relative humidity
2	1 year
2a	4 weeks
3	168 hours
4	72 hours
5	48 hours
5a	24 hours
6	Mandatory bake before use. After bake, must be reflowed within the time limit specified on the label.

While components are still in the bag as sealed by the distributor, a desiccant packet keeps them dry. Once you open the bag, the timer as specified by the MSL starts. Therefore, it's best not to open bags of components until right before you're about to use them. Many assemblers have "dry boxes," which are storage areas for parts that keep humidity very low. These can be used to store opened bags. If parts exceed the limits in the MSL table, or even if they're still sealed but very old, they need to be *baked out* before assembly: A low-temperature oven is used over hours or days to very slowly dry out parts so they aren't damaged.

Determining Quantity

Buy more than the exact number of parts you need. If you're hand-assembling, you'll inevitably drop or lose parts, especially tiny ones. Pick-and-place machines will also drop parts. Assemblers usually request that they be provided with an extra margin of parts for this purpose. Table 2-2 provides some guidelines for how many extra parts to buy, based on the type and size of the part.

Table 2-2: How Many Extra Components to Buy

Component type and size	Percent extra
Discrete SMT, 0201	50%
Discrete SMT, 0402	25%
Discrete SMT, 0603	15%
Discrete SMT, 0805	10%
Discrete SMT, 1206 or larger	10%
Active, 0.6 mm or less	20%
Active, 0.6 mm to 1 mm	10%
Active, 1 mm to 1.6 mm	5%
Active, 1.6 mm to 2 mm	5%

Component type and size	Percent extra
Active, greater than 2 mm	5%
Small through-hole devices	5%
Other through-hole devices	2%

In addition to extra electronic components, you'll also need to provide your assembler with extra mechanical fasteners like nuts and screws. Table 2-3 offers some guidance on quantities.

Table 2-3: How Many Extra Mechanical Fasteners to Buy

Component cost	Percent extra	Minimum extra
Less than $0.50	10%	5
$0.50–$2.00	5%	2
Greater than $2.00	2%	1

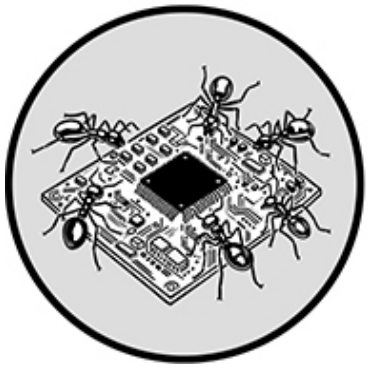
If you have really expensive parts on your BOM, you can reduce the number of extras you provide and try to recover dropped parts. However, recovering dropped parts usually isn't worth the effort given their cost versus how much your time is worth. The recovered parts would also have to be put back on tape somehow, and even identifying which part is which could be painstaking. The exception is for physically large parts and very expensive parts. For example, if your design uses a $5,000 FPGA, you aren't going to just forget about one that falls off your pick-and-place machine.

Conclusion

One of the keys to a great design can be summed up like this: *Know thy parts*. You can't meet specifications or develop or debug a design without deeply knowing your parts. Knowing your parts starts in component selection, and the decisions you make here will affect the rest of your development cycle, so take the time to choose carefully, be thorough,

and read the datasheets. Every hour you spend selecting components will save you five hours down the line. In the next chapters, we'll zoom in, with specific guidance on how to select all kinds of passive components and active components.

3
SELECTING PASSIVE COMPONENTS



Most of the components in your BOM will probably be passive. This means that they represent a substantial part of your cost and your supply chain. This chapter features guidance on selecting all kinds of passive components, including capacitors, resistors, and inductors, as well as cables, connectors, and circuit protection.

Passive parts are also sometimes known as "jellybean" components, which suggests that they're just sprinkled around your board and that less thought goes into selecting them than expensive active components like ICs. However, while passive components do typically cost much less than active components, they have a huge effect on how well your device will perform. They therefore need just as much attention to provide the proper care and feeding of your active components.

Capacitors

Capacitors are present in practically every circuit, but many designers mistakenly treat all capacitors as equal. Picking capacitors correctly can improve your design's performance and prevent insidious problems that are hard to debug. That process begins with navigating the many different species of capacitors that are available.

The three most common types are ceramic capacitors, electrolytic capacitors, and film capacitors. The Venn diagram in Figure 3-1 summarizes

appropriate applications for each capacitor type.

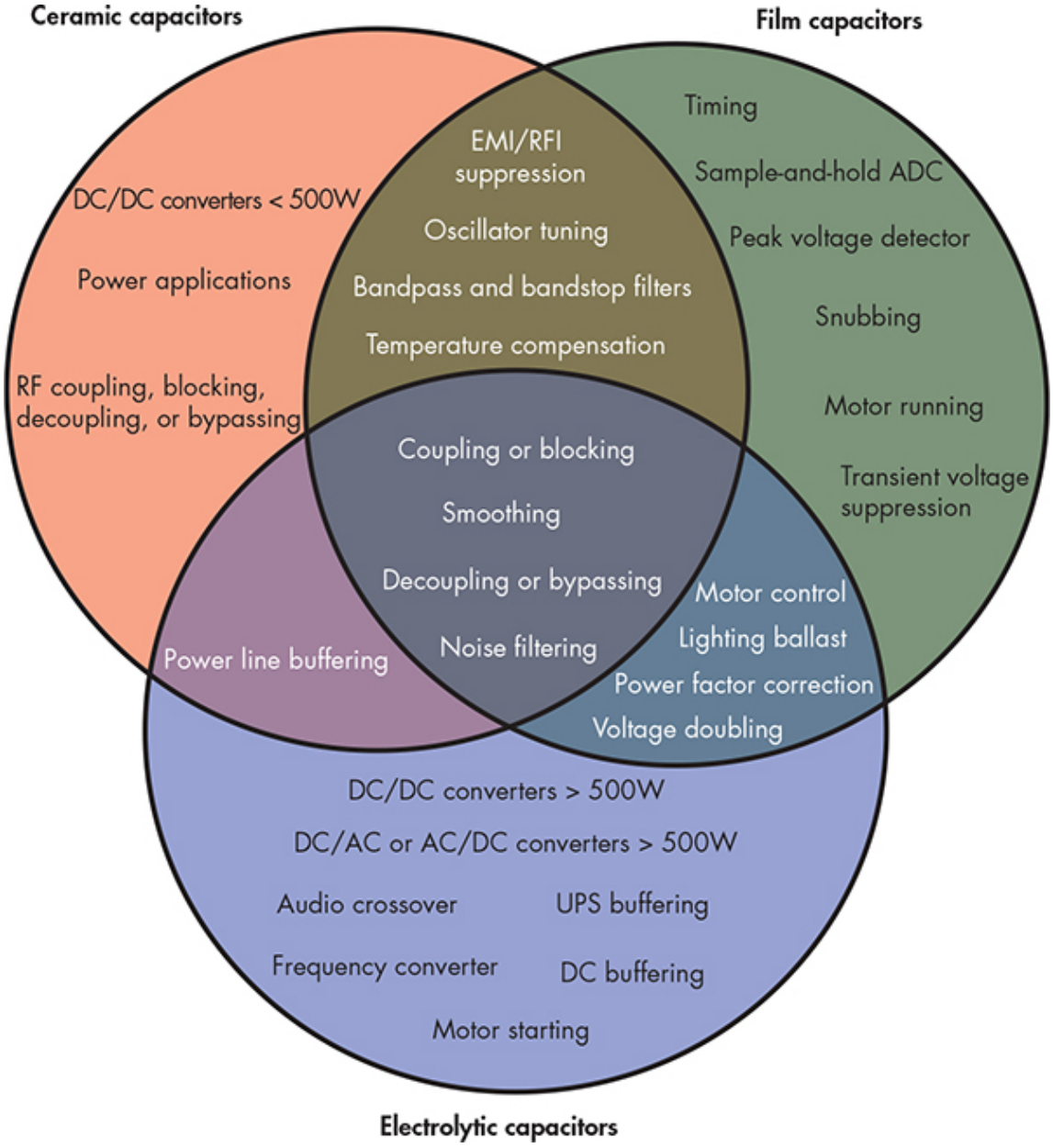


Figure 3-1: Applications of different capacitor types

Beyond these more common types, there are also supercapacitors and tantalum capacitors. We'll discuss each of these types in turn and look at some other details to keep in mind when choosing capacitors.

Multilayer Ceramic Capacitors

Multilayer ceramic capacitors (MLCCs) are perhaps the most common type of capacitor, and they are often used for bypassing, decoupling, and filtering. These capacitors are commonly just referred to as *ceramic capacitors*. They work at voltages up to about 100 V and are available up to about 50 µF. Ceramic capacitors are available with a variety of different dielectric materials. These have names like X5R, C0G, and Y5V. These codes denote the capacitor's temperature coefficient and the tolerance of that temperature coefficient. There are two classes of ceramic capacitor: Class I and Class II. Class I capacitors are what all other capacitors wish they could be: They have a linear temperature coefficient, their capacitance stays the same across voltage, they're low loss, and they have high stability and accuracy. They also have a high quality factor (Q), making them suitable for high-frequency use. Of course, there's a catch: Class I capacitors are available only in low capacitance values. You're not going to find a 100 µF Class I capacitor. In fact, you're going to have a hard time finding anything above 1 µF, and even 0.1 µF is going to be physically large. The most common Class I capacitors you'll encounter are NP0/C0G. Many other types of Class I capacitors are available, but I've gone my entire career without ever using them.

If you need higher capacitance in a small volume, you must use a material with a higher relative permittivity, which means going to a Class II capacitor. That also means compromising on all of the wonderful properties of Class I capacitors and introducing potential side effects like a piezoelectric effect. This is okay as long as you use Class II capacitors correctly. For most purposes, the only Class II capacitors you should use in your design are X5R and X7R. There are lots of other dielectrics available; however, some, like Y5R, have poorer tolerance and poorer performance over temperature, but still cost about the same. Others, like X8R, are designed for high-temperature applications and perform better than X5R and X7R.

Class II ceramic capacitors must be *derated* for your design, meaning you need to choose parts that have a higher rating than what you expect them to need. A very common novice mistake is to determine that you need, say, a 10 V and 10 µF capacitor and then go to your distributor of choice and pick out a capacitor rated for 10 V and 10 µF. This can lead to trouble for a couple of reasons. First, it's likely that the voltage on that capacitor when it's in your circuit is going to exceed 10 V some of the time. There will be transients and voltage spikes and dips that may only last milliseconds, but

that still affects the capacitor's performance. Second, as you increase the voltage on a Class II ceramic capacitor, its effective capacitance goes down. A good capacitor datasheet will include a voltage versus capacitance curve (although not all datasheets will contain this curve). What this will tell you is that sometimes even a capacitor that says it's 10 µF and 10 V won't actually give you 10 µF *at* 10 V! It might be more like 4 µF. The plot in Figure 3-2 shows some example voltage versus capacitance curves. Don't use this plot to choose parts; use the charts in the datasheet of the components you're considering.

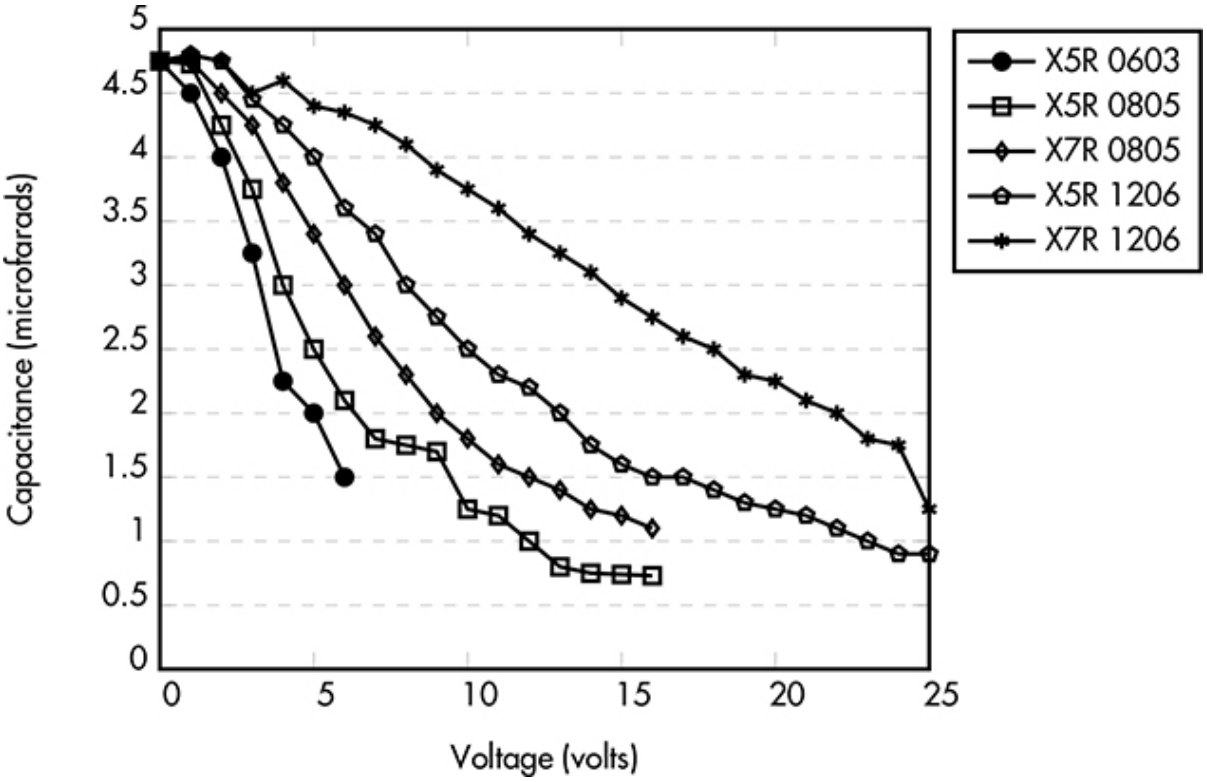


Figure 3-2: Example voltage versus capacitance curves

The voltage versus capacitance curve is why you should always pick a capacitor with a higher voltage rating than what you designed for. Table 3-1 shows the capacitor derating recommendations used by NASA. (This and other NASA guidelines are taken from MIL-STD-975, "NASA Standard Parts List.")

Table 3-1: Capacitor Derating Recommendations

Type	Derating factor	Max. ambient temperature
Ceramic	60%	110°C
Tantalum	50%	70°C
Plastic film	60%	85°C

This table, which extends to other types of capacitors besides ceramic, illustrates that if, for example, you need a tantalum capacitor at 10 V, you should actually use one rated for 15 V (50 percent higher). Feel free to use parts that exceed these derating factors. Also keep in mind that the voltage versus capacitance curve gets worse as the component's package size gets smaller. In other words, a 0402 capacitor will start losing capacitance with increasing voltage faster than a 1206 capacitor will. If you're having trouble finding a part with a good enough capacitance at your voltage, try looking at a physically larger package. More detailed derating information and modeling is available in the US military handbook MIL-HDBK-217F, *Reliability Prediction of Electronic Equipment*.

If you're curious about why capacitance changes with voltage, the answer has to do with materials science. Class II capacitors are made of barium titanate, and barium titanate molecules are slightly polarized, which makes them act like little dipoles. When the voltage across the capacitor is zero, the dipoles point in random directions and capacitance is high. As you start to raise the voltage, the dipoles start to line up, which decreases the capacitance. Physically bigger capacitors can maintain their capacitance at higher voltages because their dielectric material is thicker. This means a weaker electric field, which in turn means fewer dipoles start aligning.

Aluminum Electrolytic Capacitors

Aluminum electrolytic capacitors (sometimes referred to as just *electrolytic caps*) are usually used in circuits that need a large amount of capacitance. Electrolytic caps range from about 0.1 µF to several farads. However, they're physically much bigger than other types of capacitors. Electrolytic caps are also available at higher voltages than other kinds of capacitors. For example, you won't be able to find a 1 µF MLCC rated for 400 V (they max

out at about 100 V), but you'll be able to find electrolytic caps that can meet those requirements.

Many electrolytic caps are tall, which can make them difficult to integrate into space-constrained designs. If you need the capacitance and voltage of an electrolytic cap but can't fit one, you may be able to use a *polymer aluminum electrolytic capacitor*. They're low-profile and use a conductive polymer instead of a liquid electrolyte.

Electrolytic caps are polarized. If you power them in the reverse polarity, the electrolyte inside the capacitor will heat up, vaporize, and possibly cause the capacitor to explode. A blown capacitor can spew electrolyte everywhere, which can cause shorts on other parts of your board. Be very careful about installing electrolytic capacitors the right way. Silkscreening capacitor polarity indications onto the PCB can help avoid those kinds of mistakes.

Unlike other capacitor types, electrolytic capacitors have a shelf life. This shelf life is usually two to three years, assuming the capacitors have been stored correctly. Check with the manufacturer for the shelf life for specific part numbers. It's possible to "reform" electrolytic capacitors if they've been stored beyond their shelf life to get them back to near their original performance. The US military has a handbook on how to do this called MILHDBK-1131, *Storage Shelf Life and Reforming Procedures for Aluminum Electrolytic Fixed Capacitors*.

Electrolytic capacitors are prone to drying out over time, which makes them poorly suited to hot operating environments. If you know your board will experience high temperatures during its lifetime in the product, it's a good idea to avoid electrolytic caps.

Film Capacitors

Film capacitors are great for applications where you need low equivalent series resistance (ESR) and low equivalent series inductance (ESL—the *L* is the traditional designator for an inductor). They also have great temperature stability and aren't polarized. These capacitors are commonly made with polyester film, but if you need film capacitors with an even higher temperature stability and are willing to pay more for them, look for polypropylene film capacitors instead of polyester.

Film capacitors can handle higher current surges than electrolytic capacitors, but they're more expensive and physically bigger than electrolytic caps. You'll have a hard time finding a lot of film capacitors in an SMT package. Good applications for film capacitors include circuits that require a strong frequency stability or high Q, like resonant circuits. They're also great for applications that require handling high voltage surges, like snubbers.

Devices that run on AC power sometimes use a special kind of film capacitor for filtering and reducing electromagnetic interference from the AC line. These capacitors are available in Class X and Class Y versions. Class X capacitors are placed between the line and neutral wires, so they're designed to fail short and trigger a fuse to blow or circuit breaker to trip. Class Y capacitors are used between the line and ground connections, so they're designed to fail open to prevent a dangerous ground fault, which could create an electric shock hazard. There are subtypes of Class X and Y AC line filter caps that have different voltage ratings. No matter which specific AC line filter capacitor you choose, make sure that it's safety certified by a reputable certification organization.

Supercapacitors

Supercapacitors are a relatively recent entry to the capacitor market. The most common kind are electric double-layer capacitors (ELDCs). These caps have very low voltages (around 2.7 V) but extremely high capacitance, up to hundreds of farads. ELDCs are polarized, but reverse biasing *symmetric* ELDCs doesn't cause them to fail catastrophically. However, it does cause them to degrade, so you should avoid it. ELDCs that are marked *asymmetric* can't be safely reverse biased.

While all ELDCs have a large capacitance, their low voltage means that the energy stored in them isn't as huge as you might initially think. Many applications that use ELDCs could also use a small lithium battery, but they use a supercapacitor instead because of the increased lifetime and possibly the higher current surge capability. Supercapacitors also have a very low self-discharge rate because they have such a low internal resistance. For this reason, supercapacitors are often seen in circuits that require a small battery backup to keep a clock running or volatile memory alive while the system is powered off.

Tantalum Capacitors

Tantalum capacitors are mostly used in applications needing a large amount of capacitance in a small volume. However, be very cautious with them because although they do maintain their capacitance over voltage, tantalum capacitors can fail catastrophically if exposed to a voltage above their rating. This means it's necessary to derate them well below their advertised voltage (see Table 3-1). If you must use tantalum, use polymer tantalum, not manganese dioxide tantalum. Polymer tantalum capacitors will still fail in an overvoltage condition, but they won't explode. Polymer tantalum capacitors can also be derated a little less conservatively—say, by 20 percent.

Tantalum capacitors should only be sourced from a Tier 1 supplier (that is, a company that directly manufactures them) with a documented, conflict-free supply chain. Tantalum is considered a conflict mineral, and it's important to only use components that have ethically sourced materials.

Capacitance vs. Frequency

An important detail in a capacitor datasheet is the capacitance versus frequency curve. There's no such thing as an ideal capacitor; as you start cranking up the frequency of a signal into a capacitor, it will start looking less like a capacitor and more like an inductor. The point at which a capacitor looks more like an inductor is its *self-resonant frequency (SRF)*. Make sure you're using all capacitors below their SRF. Figure 3-3 shows an example of the frequency response of a couple of capacitors.

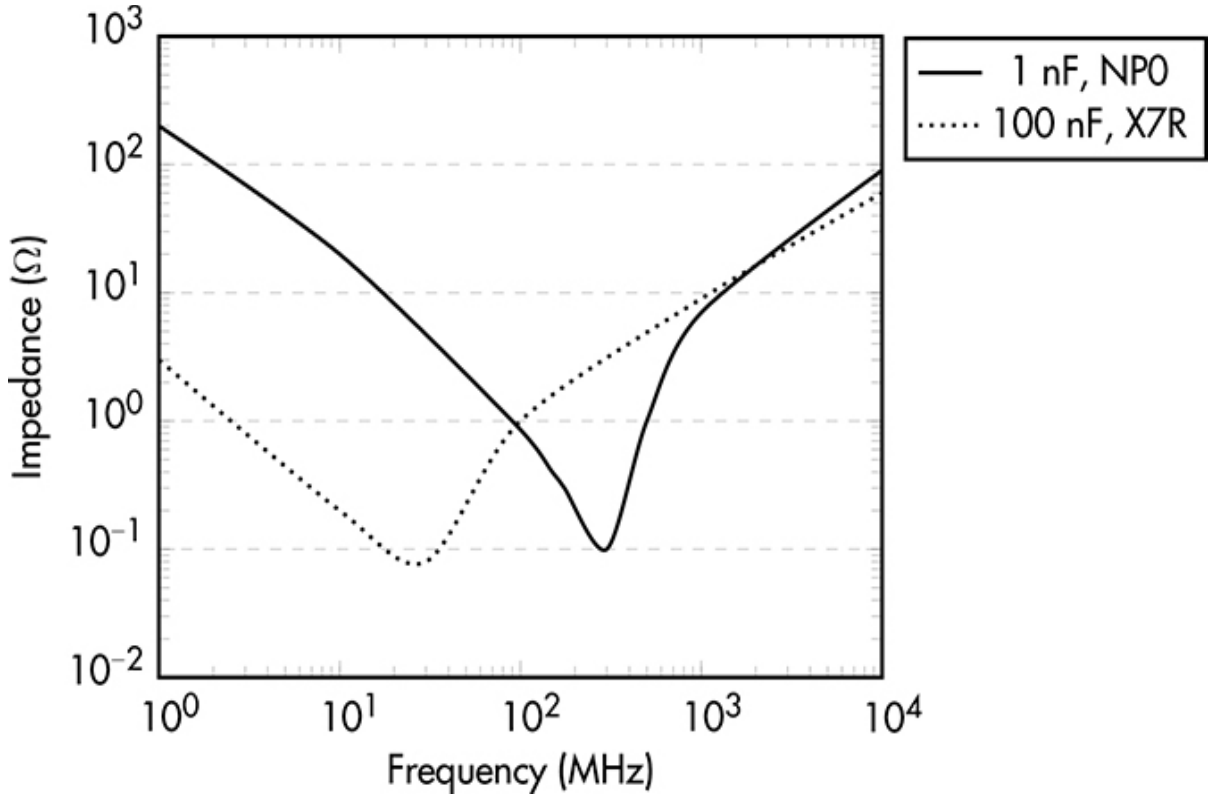


Figure 3-3: Example plot of capacitor self-resonant frequency

In the figure, the SRF is the lowest point of each plot. All points to the left look more capacitive, while all points to the right look more inductive. You can see how the size and rating of the capacitor changes the location of the SRF. Generally, the higher the capacitance and the bigger the physical size, the lower the SRF.

The capacitance versus frequency curve also indicates a capacitor's equivalent series resistance. This appears in the plot as the distance above zero on the y-axis. You should almost always make sure the ESR of your capacitors is low. A low ESR is important to prevent power and heating losses. It also improves the response time of capacitors to voltage transients. You'll sometimes see a callout in a datasheet or application note that low-ESR capacitors must be used.

There are some occasions when you might not want to use capacitors with as low an ESR as possible. Sometimes it's helpful to actually add some resistance. To understand why, think about the non-idealized model of a capacitor shown in Figure 3-4.

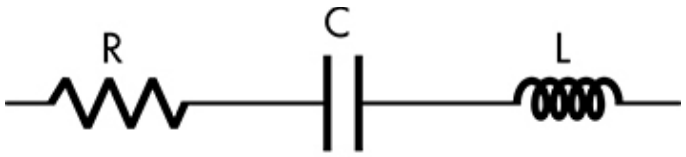


Figure 3-4: A non-ideal model of a capacitor

It's an RLC circuit, and changing the resistance in an RLC circuit will change its Q. This creates a trade-off: If you're willing to accept slightly worse performance (higher impedance) in some areas, you can get better performance (lower impedance) in other areas. The best way to see this is to look at an example. Figure 3-5 plots the impedance levels of two identical capacitor networks on top of each other. The only difference between them is that the one plotted with a dotted line uses low-ESR capacitors, while the one plotted with a solid line uses a combination of low-ESR and controlled-ESR capacitors.

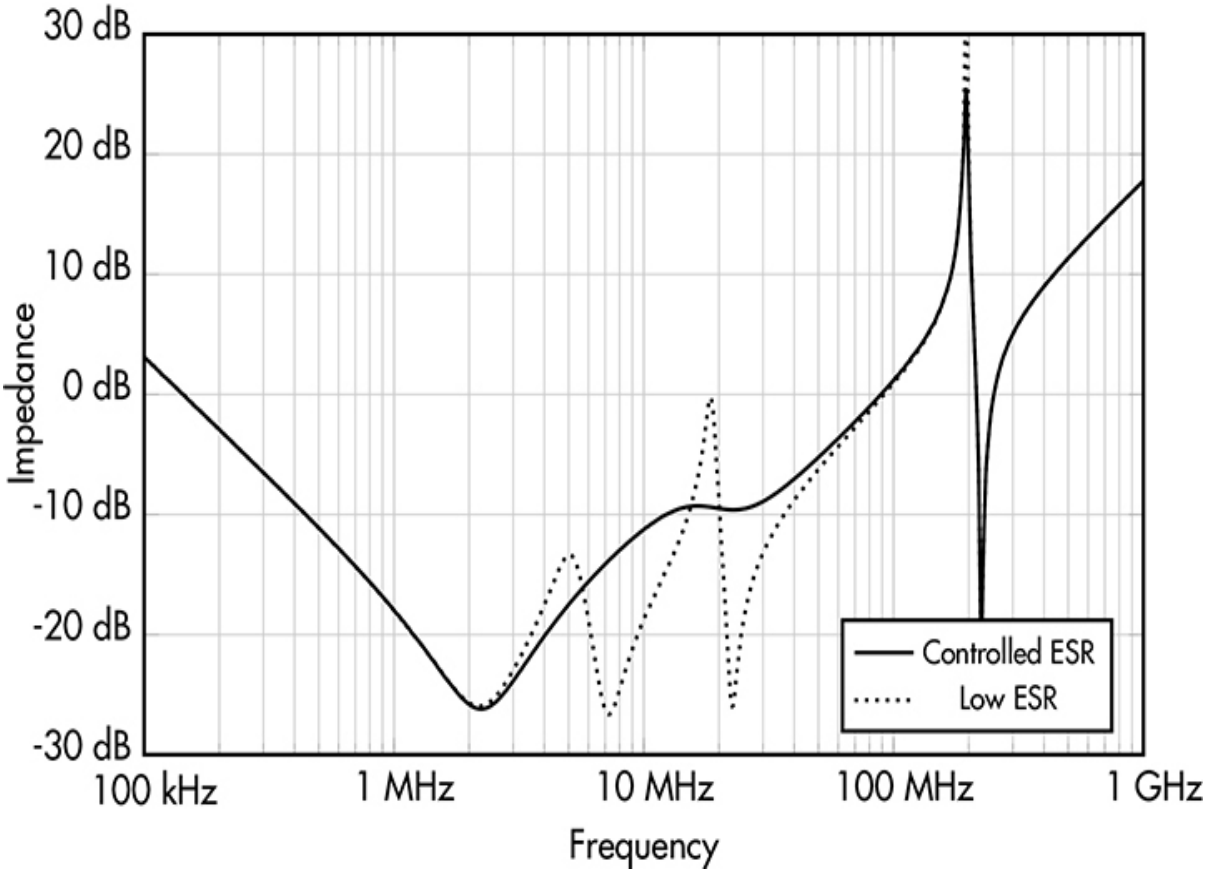


Figure 3-5: A capacitor network with low ESR versus controlled ESR

You can see that using just low-ESR capacitors results in a much lower impedance for many frequencies, but it's not very smooth. Adding in some series resistance on purpose smooths out the response and even achieves a lower impedance in some specific areas. Controlled-ESR capacitors still have low resistance: In this simulation, I used 0.75 Ω as the controlled ESR and 0.05 Ω as the low ESR. But when you know you need better capacitor performance at certain frequencies, using controlled-ESR capacitors can help you get there. The process of figuring that out is called *power distribution network (PDN) design*, and it's an entire subfield of electrical engineering. In general, you really only need to worry about it if you're designing with devices like processors or FPGAs. Small, low-power devices like a little microcontroller will work just fine if you follow the recommended decoupling network in the datasheet.

Some low-dropout (LDO) regulators have a minimum ESR requirement on the output capacitors to prevent oscillation. Your first thought may be to use a ceramic capacitor for this job, but the ESR requirement that some LDOs have may not be achievable with ceramic capacitors. You'll need to choose a different capacitor type. Watch out for this!

High-Frequency Signals

High-speed and RF signals require high-Q capacitors. If you use a regular, non-high-Q capacitor, bad things will happen: Your filters won't have the response you expect, your amplifiers will oscillate, and you'll see very high insertion loss through your capacitors. Capacitor manufacturers have specific product lines for high-frequency use, and it's critical that you choose capacitors that are rated for the frequency you're using. For example, the GJM capacitor series by Murata (their high-Q series) has a self-resonant frequency of 16 GHz. Johanson is another good manufacturer of high-Q capacitors.

Think carefully about where the high-frequency signals in your design are and make sure that you call out the right part numbers in your BOM to prevent signal integrity issues. If you don't do any RF engineering, you may think that you'll never encounter this problem. However, many wireless radio ICs (Bluetooth, Wi-Fi, NFC, and so on) require a small matching network or a balun right before the antenna that uses discrete inductors and capacitors. Bluetooth and Wi-Fi both operate at 2.4 GHz, so if you don't

use high-Q components, your radio IC that's supposed to work out of the box may not work at all.

Parasitic Inductance

The parasitic inductance (also known as ESL) of a capacitor is relatively independent of the capacitance. Instead, it depends mostly on the package size. As you increase the package size, the inductance also increases. Table 3-2 lists the approximate parasitic inductance of different ceramic and tantalum capacitor package sizes, from "Comparison of Multilayer Ceramic and Tantalum Capacitors" by Jeffrey Cain.

Table 3-2: Capacitor Stray Inductance

Capacitor size	Inductance (pH)	Type
0603	850	Ceramic
0805	1,050	Ceramic
1206	1,250	Ceramic
1210	1,020	Ceramic
0805	1,600	Tantalum
1206	2,200	Tantalum
1210	2,250	Tantalum
2312	2,800	Tantalum

Since capacitors that are specifically rated to be high-Q are designed to be used at high frequencies, you don't need to worry about package inductance. It's already taken into account. As a result, you'll be hard pressed to find high-Q capacitors in packages bigger than 0603.

Common Capacitor Values

Like other passive components, capacitors are commonly produced with values in discrete intervals. Those intervals are defined by the *E series*, a standard set of values that all component manufacturers have agreed to produce. This makes sourcing components easier on designers. There are different sets of values available depending on the tolerance of the

component you need. For example, Table 3-3 shows the E12 series, which corresponds to parts with a 10 percent tolerance and has 12 different values per decade in it.

Table 3-3: Standard E12 Series Capacitor Values

10 pF	100 pF	1000 pF	.010 µF	.10 µF	1.0 µF	10 µF
12 pF	120 pF	1200 pF	.012 µF	.12 µF	1.2 µF	
15 pF	150 pF	1500 pF	.015 µF	.15 µF	1.5 µF	
18 pF	180 pF	1800 pF	.018 µF	.18 µF	1.8 µF	
22 pF	220 pF	2200 pF	.022 µF	.22 µF	2.2 µF	22 µF
27 pF	270 pF	2700 pF	.027 µF	.27 µF	2.7 µF	
33 pF	330 pF	3300 pF	.033 µF	.33 µF	3.3 µF	33 µF
39 pF	390 pF	3900 pF	.039 µF	.39 µF	3.9 µF	
47 pF	470 pF	4700 pF	.047 µF	.47 µF	4.7 µF	47 µF
56 pF	560 pF	5600 pF	.056 µF	.56 µF	5.6 µF	
68 pF	680 pF	6800 pF	.068 µF	.68 µF	6.8 µF	
82 pF	820 pF	8200 pF	.082 µF	.82 µF	8.2 µF	

If you buy a capacitor kit, these are usually the values that it comes with. Rounding your capacitor values to one of these common values (when you can safely do it) will make it a little easier to prototype and troubleshoot, since you'll probably have those values on hand.

The complete E series includes E3, E6, E12, E24, E48, E96, and E192. The E3 series has only 3 values per decade, representing a tolerance of 40 percent, and E192 has 192 values per decade, representing a tolerance of 0.5 percent.

Fail-Safe Capacitors

Some capacitors are designed to be fail-safe. For example, while ordinary tantalum capacitors can fail short if they're exposed to overvoltage conditions, some tantalum capacitors have built-in protection to combat this. They're called either *fail-open* or *fused* tantalum capacitors. These

capacitors have the advantage of failing as an open circuit instead of a short circuit and thus can prevent further damage in your circuit. The disadvantage is that the fuse adds extra ESR, so keep this in mind if using a fused capacitor.

MLCCs aren't vulnerable to catastrophic failure in an overvoltage condition like tantalum capacitors are, but they're vulnerable to internal cracking if the PCB they're soldered to is flexed. As the name implies, MLCCs have multiple layers of electrodes inside them that are stacked on top of each other, with a ceramic dielectric sandwiched between each layer. Ceramic is brittle, and if it cracks due to stress, adjacent electrodes can touch. To prevent these internal cracks from causing a short, you can use *open-mode* or *floating-electrode* MLCCs. These work by separating the electrodes horizontally so that a crack will be less likely to short out both sides of the capacitor. Figure 3-6 shows the difference.

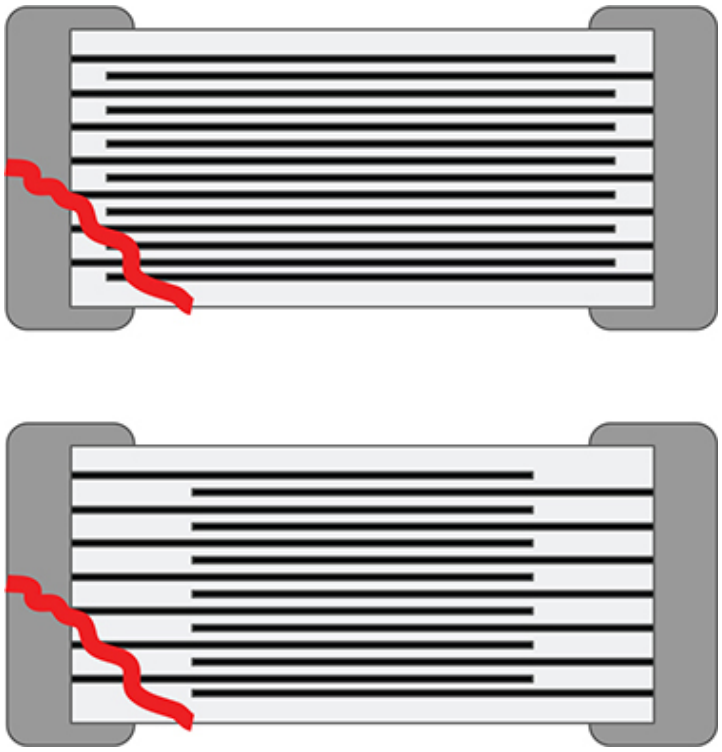


Figure 3-6: A crack in a typical MLCC (top) can cause adjacent electrodes to short together. An open-mode MLCC (bottom) has a lower chance of causing a dead short when cracked because the electrodes are horizontally spaced.

For even more safety, you can use a *flexible termination capacitor*, which further reduces the chance of an internal short. These work by changing the way the terminals of the capacitor are connected to the metal solder points

so that stress isn't transferred to the interior of the capacitor in the first place. This is usually done with a conductive polymer layer on the terminations. An open-mode capacitor can withstand PCB flexing of up to about 2 mm, whereas a flexible termination capacitor can withstand flexing of up to about 5 mm. These numbers are just guidelines, so make sure to read the datasheet of any fail-safe capacitor before you use it.

Resistors

Resistors are arguably the cheapest and simplest components you'll work with. They're used for a wide range of applications, and along with capacitors, you're almost certain to use at least one in even the most basic circuits. Let's talk about how to use them wisely.

Tolerance

Choose resistors that have the tolerance your application requires. There's no need to go overboard when precision isn't important. For example, you don't need to use 1 percent tolerances for your pull-up resistors because the exact value there doesn't really matter. Since tolerances are specified as a percentage, the tolerance you need will change based on the value of the resistor you're selecting. For a very small value current-sense resistor, a 0.5 percent, 1 percent, or 5 percent tolerance may all be acceptable simply because the value of that resistor is so small that even 5 percent is low enough not to matter.

ESD Damage

We usually think of ICs and other active components as being vulnerable to ESD, but some resistors can be damaged by an ESD event as well. Both thin- and thick-film SMT resistors can be affected by less than 5 kV of ESD because the films are so thin. This can manifest itself in a few ways: Either your resistor turns into an open circuit or the resistance can change. Sometimes latent damage can occur and one of those things will happen in the near future. To prevent these problems, follow standard ESD precautions in your design and when handling components. You can also use a foil resistor, which is less susceptible to ESD.

Temperature Effects

Resistors change their resistance based on their temperature. The number that describes this is called the *temperature coefficient of resistance (TCR)*. Resistance has a nonlinear relationship with temperature and can either increase or decrease as temperature goes up or down, depending on what material the resistor is made of. For most designs, resistors likely won't experience enough temperature change for this to be a problem. It does matter for some circuits, though, so it's still important to keep in mind.

Another important thing to remember is that different kinds of resistors have different amounts of thermal noise. This is especially important when you're using a resistor as a sensor. The rule of thumb is that thick-film resistors have higher thermal noise than thin-film resistors, and physically larger resistors have higher thermal noise than physically smaller resistors.

High-Power Resistors

Resistors that can handle high power are physically large and sometimes require a heat sink. Plan accordingly. It's best to try to design your circuit such that you don't need to use high-power resistors that are burning up power; it's wasteful, and the resistors are expensive. If your resistors will be subjected only to quick pulses of high power or voltage, you can look for resistors labeled "pulse withstanding."

If a resistor will be carrying a large amount of current or a high voltage, calculate the power that will be dissipated through it. NASA recommends derating resistors to 60 percent of their rated power. Make sure that any heat released won't affect any other part of your circuit and is dissipated correctly.

Current-Sense Resistors

One job that resistors sometimes do is help you measure current. By measuring the voltage drop across a resistor, you can calculate the current through it using Ohm's law. To do this practically, your resistor must have several qualities:

- A very small resistance value, so you're not burning up lots of power
- A precisely known resistance that remains stable
- The ability to handle potentially high currents

Current-sense resistors are designed with these requirements in mind. They often come in the milliohm range, have very high tolerance, and are physically big enough to be able to dissipate whatever power gets burned up in them without changing resistance. Some current-sense resistors also come in four-terminal versions to allow you to use the Kelvin method for improved accuracy.

Resistors as Voltage Dividers

When picking resistors for a voltage divider, or for pulling up or pulling down a signal, consider the current leakage through that resistor. Using a resistor value between 10 kΩ and 100 kΩ will keep your design from drawing unnecessary power. Calculate the optimal resistor values that will provide enough current for your device to function, but not so much that you're burning up a lot of excess current as heat in the resistors.

Don't use a voltage divider to power a circuit directly. Instead, use a more efficient method that can actually regulate the output voltage, like an LDO or a switched-mode power supply. It's also not a great idea to use a voltage divider to convert logic levels. It's possible to use a voltage divider to convert a 5 V signal to a 3.3 V signal, for example, but there are a couple of reasons why you shouldn't do this. First, for any signals with fast edges or a high frequency, the stray inductance and capacitance of the resistors will wreak havoc on your signal integrity. Second, a resistor divider won't work for a bidirectional signal because it can only drop voltage. Your 5 V signal can be seen by your 3.3 V device, but your 3.3 V signal may not be seen by your 5 V device. However, for slow signals, like enable/disable lines, a voltage divider will work for converting higher-voltage signals to lower-voltage signals (again, only in one direction).

Common Resistor Values

As with capacitors, resistors are available in standard values. Table 3-4 shows the intervals in the E24 series. Note that resistors sold in these intervals are typically 5 percent tolerance.

Table 3-4: Standard Resistor Values

1.0	10	100	1.0K	10K	100K	1.0M
1.1	11	110	1.1K	11K	110K	1.1M

1.2	12	120	1.2K	12K	120K	1.2M
1.3	13	130	1.3K	13K	130K	1.3M
1.5	15	150	1.5K	15K	150K	1.5M
1.6	16	160	1.6K	16K	160K	1.6M
1.8	18	180	1.8K	18K	180K	1.8M
2.0	20	200	2.0K	20K	200K	2.0M
2.2	22	220	2.2K	22K	220K	2.2M
2.4	24	240	2.4K	24K	240K	2.4M
2.7	27	270	2.7K	27K	270K	2.7M
3.0	30	300	3.0K	30K	300K	3.0M
3.3	33	330	3.3K	33K	330K	3.3M
3.6	36	360	3.6K	36K	360K	3.6M
3.9	39	390	3.9K	39K	390K	3.9M
4.3	43	430	4.3K	43K	430K	4.3M
4.7	47	470	4.7K	47K	470K	4.7M
5.1	51	510	5.1K	51K	510K	5.1M
5.6	56	560	5.6K	56K	560K	5.6M
6.2	62	620	6.2K	62K	620K	6.2M
6.8	68	680	6.8K	68K	680K	6.8M
7.5	75	750	7.5K	75K	750K	7.5M
8.2	82	820	8.2K	82K	820K	8.2M
9.1	91	910	9.1K	91K	910K	9.1M

Of course, resistors are available in other values as well. There are even such things as 0 Ω resistors; they're often used as jumpers, as test points, or as a way to short-circuit another series component. However, 0 Ω resistors can sometimes have an actual value as high as 1 Ω. If you need something

very close to 0 Ω, it's better to buy a low-value, high-tolerance part like a 0.001 Ω resistor with a 1 percent tolerance.

If you're doing RF work and looking for a 50 Ω resistor, you'll find that there are startlingly few options available, and they're all very expensive. Instead, look for 49.9 Ω resistors. The difference of 0.1 Ω is negligible, and you'll have a much easier time finding the part you need, since many manufacturers make a 49.9 Ω resistor as part of the E96 series but not a 50 Ω resistor. A 50 Ω resistor is technically a special value, so manufacturers will charge much more for it.

Inductors

Inductors find use in applications like switching power supplies and RF circuits. NASA recommends derating inductors to 50 percent of their rated voltage and 60 percent of their rated temperature. Like capacitors, inductors should have a low ESR and must be operated below their SRF. Check the curves in the datasheet (if they're present). Figure 3-7 shows an example.

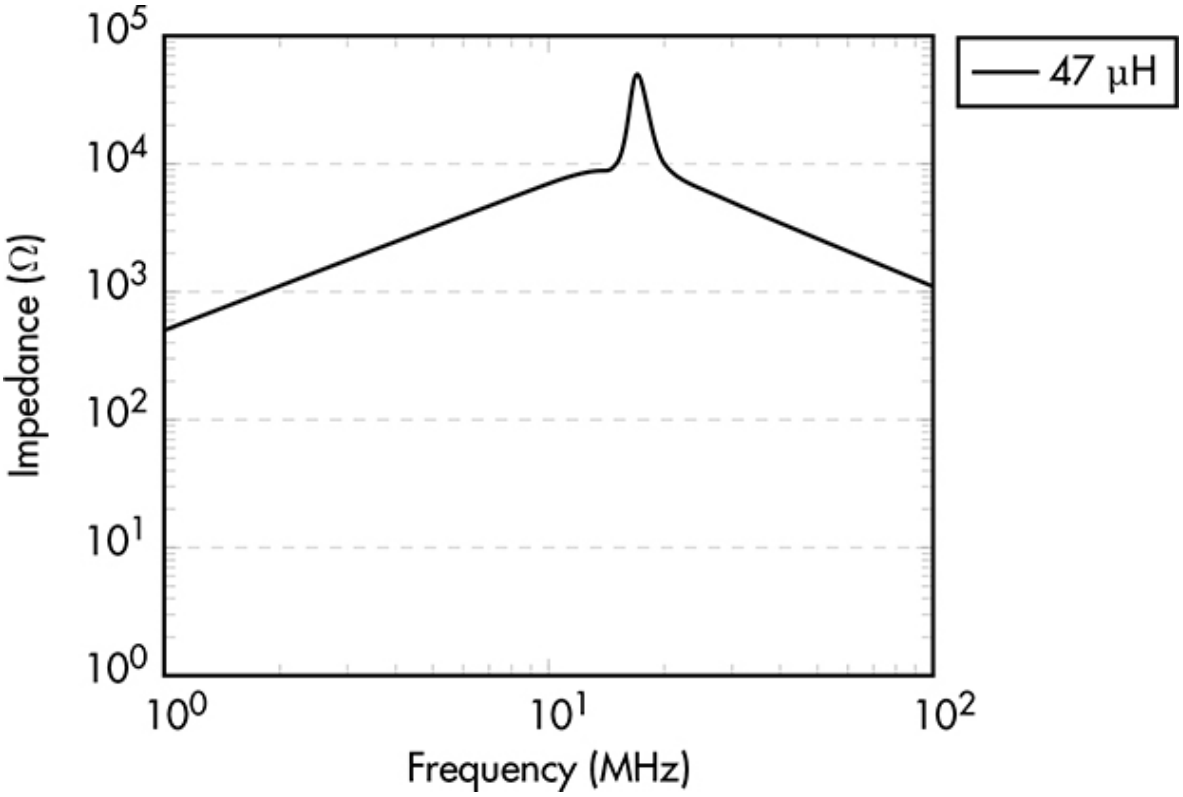


Figure 3-7: Example inductor self-resonant frequency

The SRF of the example inductor is indicated by the peak in the curve. At all frequencies lower than that, the inductor looks mostly like an inductor, and at all frequencies higher than that, it looks like a capacitor. Stick to the frequency range where your inductor looks inductive. Notice that this graph is the inverse of the capacitor SRF graph shown in Figure 3-3.

Switched-Mode Power Supplies

Inductors are often used in switched-mode power supplies (SMPS). An inductor in an SMPS will see lots of time-varying currents as the field-effect transistor (FET) switches. This can cause electromagnetic interference (EMI) and electromagnetic compatibility (EMC) problems during compliance testing. Use a shielded inductor to help prevent interference with other components in your design and to help pass EMC testing. A shielded inductor incorporates material into the package to contain the fields produced during switching.

The trade-off here is that a shielded inductor will have a lower maximum current than the same nonshielded inductor since the core will experience flux saturation earlier. *Flux saturation* is when the material that makes up the core of the inductor is unable to hold any more energy in the form of a magnetic field—that is, magnetic flux can no longer increase. If this happens in your circuit, your inductor won't act like an inductor anymore.

The exact type of SMPS you use and the frequency at which it operates will, in part, define the amount of inductance that the circuit needs. The more inductance, the bigger the physical size of the inductor. If you're designing a device that needs to be relatively small, you're going to want a small inductor. Generally, a faster SMPS switching frequency will result in a smaller inductor and a small output ripple, but slightly worse efficiency.

High-Frequency Applications

Inductors used for high-frequency applications must be high-Q and rated for high-frequency use, much like capacitors. It's critical to choose high-Q inductors for the parts of your circuit that are high-frequency, or nothing will have the response you expect and your amplifiers will oscillate. Air-core spring-type inductors usually have a higher Q than ferrite-core inductors.

Ferrite Beads

If an inductor has a very poor Q, it becomes a different component: a *ferrite bead*. Above a certain frequency, a ferrite bead behaves like an inductor that's bad at storing energy in its magnetic field and instead dissipates most of the energy as heat. This is useful when you want to filter out high frequencies from a signal.

But what's the best way to choose a ferrite bead? Some are designed for specific applications, like USB. In that case, use the ferrite beads specific to your application. For most applications, though, the rule of thumb is to pick a ferrite bead that will filter out everything past the fifth harmonic of the signal of interest. For example, if your signal of interest is 1 MHz, use a ferrite bead that looks largely resistive starting at about 5 MHz. If you filter out lower-order harmonics, you'll start to distort your signal by increasing its rise time. Filtering at the fifth harmonic is a good balance; it will cause a square wave to have minimal distortion but still remove higher-frequency signals.

When shopping for a ferrite bead on an electronics distributor's site, you'll see high-level specifications like "120 Ω at 100 MHz." These rough specifications can often mislead you into picking a part with an inductive region that ends at a higher frequency than you think. This is a problem, since if you use a ferrite bead in its inductive region, you'll get ringing as it couples with parallel capacitance in your circuit. The only way to correctly pick a bead is to look at the frequency response plot in the datasheet, like the example plot shown in Figure 3-8, and see where resistance starts dominating inductance.

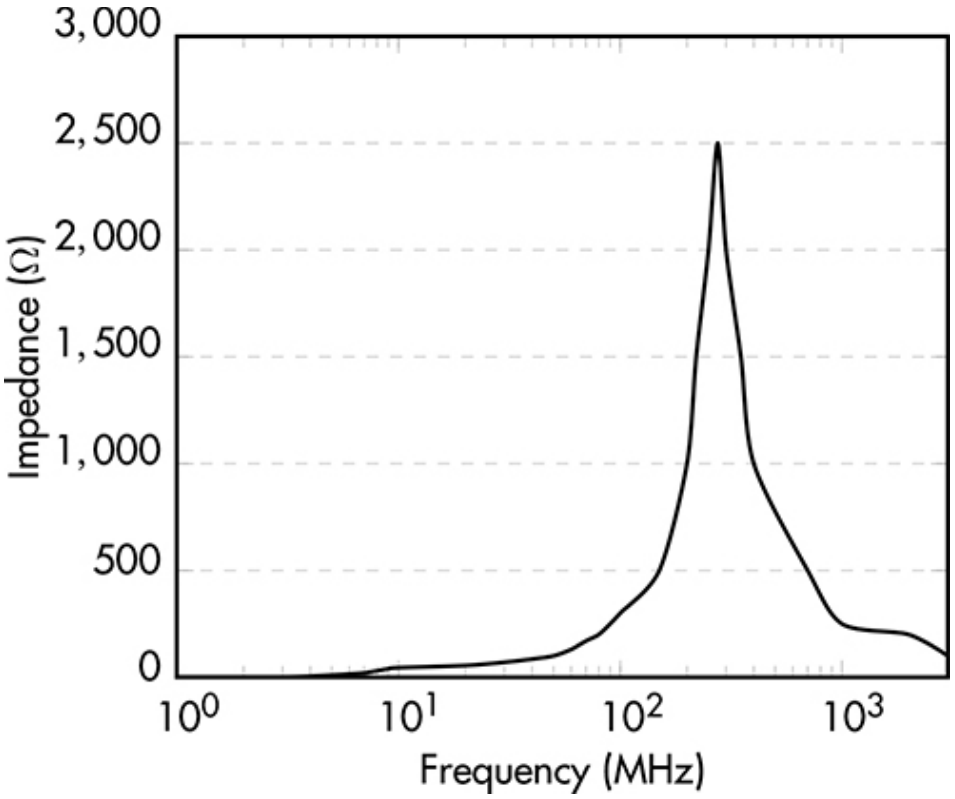


Figure 3-8: Example ferrite bead frequency response

The frequency response plot tells us that this particular bead would be useful for suppressing noise in the range of about 180 MHz to 500 MHz. Below 180 MHz, the bead looks mostly inductive, and above 500 MHz, it looks mostly capacitive. Within the band of 180 MHz to 500 MHz, it looks largely resistive, so it will dissipate energy at those frequencies as heat rather than storing it as an electric or magnetic field. That's exactly what we want. Since our goal is to filter frequencies past the fifth harmonic of the signal of interest, the fundamental signals that this bead could be used for range from 36 MHz (one-fifth of 180 MHz) to 100 MHz (one-fifth of 500 MHz).

Power Ratings vs. Signal Ratings

Ferrite beads can be rated for power or for signals. These ratings are essentially a measure of their ESR and power dissipation ability. If you're trying to filter power using a ferrite bead that's only rated for signals, the bead might fail or cause degraded performance. As with inductors and capacitors, ESR needs to be sufficiently small. Ferrite beads are often used in series with power supply pins, so a high ESR can cause an undervoltage on the device being powered.

When deciding on the maximum allowable ESR for a ferrite bead, use the maximum current that can flow through the bead, not the nominal current. If an IC nominally draws 100 mA but can surge up to 200 mA, calculate the voltage drop across the ferrite bead with 200 mA of current, not 100 mA. Otherwise, your IC will be starved for voltage during current surges.

Temperature Effects

Ferrite beads have another "feature" that you need to watch out for if your device needs to operate at high temperatures: The maximum current through a ferrite bead drops off dramatically once the temperature begins to rise. This will be different for every bead, so look at the current versus temperature curve to ensure that you won't exceed the maximum current of the bead at the maximum temperature of your design specification.

Current Effects

The amount of current through a ferrite bead affects its frequency response. The higher the current through the bead, the farther up in frequency the impedance plot will shift and the lower the maximum impedance will be. For example, a ferrite bead with 100 mA of DC current through it will have a much higher maximum impedance than the exact same bead with 200 mA of DC current running through it. Additionally, the maximum impedance point will be at a higher frequency when running 200 mA through the bead rather than 100 mA. Again, check the plots in the datasheet for exact values, but beware that a lot of plots in the datasheets will show impedance with zero DC bias current, a situation that will never happen in real life.

In general, it's a good idea to derate the maximum current limit in a ferrite bead by a factor of five. You should basically ignore the advertised maximum current rating and the resistance rating (for example, 120 Ω at 100 MHz) of ferrite beads and look at the plots in the datasheet to find the curves for the DC bias current you'll be applying before choosing a part. Small changes in current can alter a bead's performance by almost an order of magnitude in some cases.

This whole process may seem like it's hard or complicated to get right, but it's usually not. As long as you pick your bead correctly, any oscillation will most likely be damped by parasitic resistance. If you're unsure, or

suspect you may have problems with oscillation, run a quick simulation or take a measurement. Trial and error with a couple of different values is very common. We'll return to the subject of ferrite beads in Chapter 5 when we discuss schematic design.

Connectors and Cables

You may be surprised to find an entire section of this book dedicated to connectors and cables, but they can have as much of an impact on your design as any other component. Choosing the incorrect cable or connector, or placing either of them poorly, can cause your design to fail emissions tests, conduct noise into your circuit, or even cause a safety issue.

But before we look too closely at cables and connectors, here's something to consider: If you can avoid using a connector, do it. Connectors and cables add cost and can cause EMI and EMC problems. One way to cut down on connectors is to use hot-bar soldering. Lots of cheap consumer products do this to attach long ribbon cables to the PCB without requiring connectors. The stripped ends of the cable are laid down on top of pads on the PCB, and a long, hot bar is laid across the cable and PCB pads so that it heats up the entire length of one end of the cable, soldering the whole cable at once. This technique is frequently used to save cost during large production runs. Another way to connect a large number of wires without connectors is to simply have an assembler solder each wire manually. This takes longer, but some mass-manufactured products do it. The downside of both of these approaches is that they can make products a little harder to repair, can introduce reliability problems, and in some cases can be more expensive than using connectors. You'll need to determine whether it's worth it given your production volumes, product requirements, and labor costs.

Another less common way to cut down on connectors is to use card edge connectors. This way, instead of two connectors and a cable assembly, all you need is a single connector that slides into exposed contacts on a mating PCB. Card edge connectors are commonly used in modular devices and equipment. They're fine for power, ground, and slow signals, but they should be used only for fast-edge, high-speed, or RF applications if the card edge connector is specifically designed for that purpose—think Small Form-Factor Pluggable (SFP) or Peripheral Component Interconnect Express

(PCIe). A card edge connector may also require some extra mechanical engineering or a specific PCB thickness to fit correctly. For example, it may require a beveled edge and a particular tolerance, and it will nearly always need a plating of hard gold with nickel underneath. Don't underestimate the work here: You are, after all, literally designing half of a connector!

For the cables and connectors that you do need, aim for high quality. Good cables and connectors are expensive, but they can be well worth the extra cost, since lower-quality cables and connectors are often some of the first things to fail in a product. In general, cables, connectors, and electromechanical parts like switches and knobs should be qualified before you ship with them, so before you decide that you can get away with the cheapest cable you can find, order some and test them yourself. What you can get away with will depend on your application. For example, if the cable will be plugged in once and then hidden away inside the enclosure, your qualification requirements will be different than a battery connector that the user will plug and unplug hundreds or thousands of times.

Connector Requirements

Make sure any connectors you use that will carry power are rated to carry that power. NASA recommends derating the advertised maximum connector current by 50 percent. If you need a physically smaller connector, you can connect a single power net to multiple pins of a connector to reduce the current in any single wire. Be sure to also pick your connectors to withstand the maximum temperature they'll experience during reflow (if they will experience it), testing, and normal use.

Edge-mount connectors like end-launch SMA connectors must be selected to fit your board thickness (see Figure 3-9). Most PCBs are 62 mils thick by default, so this is the most common fit for edge-mount connectors, although they do come in other sizes as well. Ultimately, if you want to use this style of connector, it's best to try to keep your PCB thickness to the standard 62 mils. If you use a stackup with a different thickness, it may be difficult to find a connector that you can use, since they're only manufactured in a few discrete sizes.

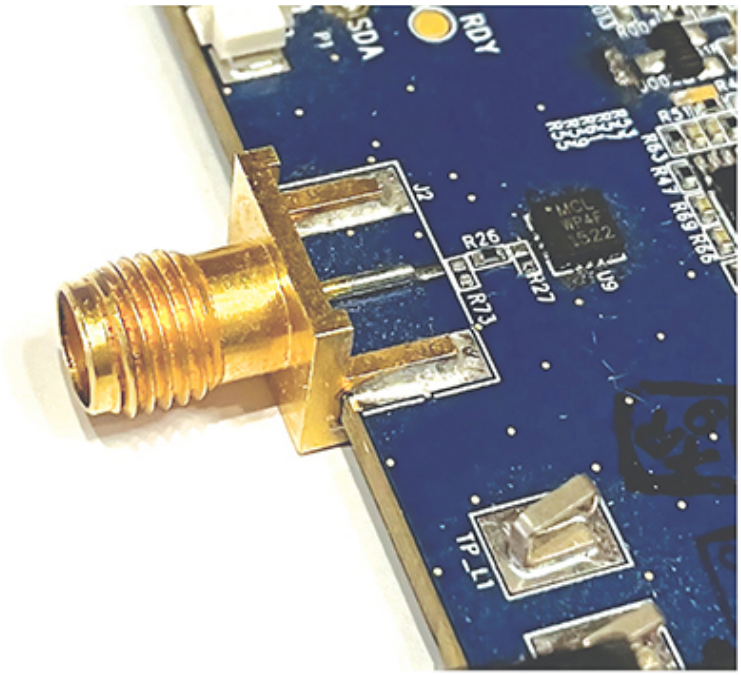


Figure 3-9: An edge-mount SMA connector

Flat flex cables (FFC) can be very useful for low-profile board-to-board connections. One trick that works well is to get an FFC that has conductors on only one side and a receptacle that has contacts on both sides. This way, you can position the receptacles on either the top or the bottom of either PCB. It also lets you bend and crease the FFC without having to change the side of the PCB that the receptacle is on. Be careful that you buy an FFC cable with the conductors on the side you intend! Off-the-shelf FFC cables can have both sets of conductors on the same side of the cable or on opposite sides of the cable, which would mirror the pinout and potentially cause problems.

Connectors should be keyed so it isn't possible to plug anything into them the wrong way. It can be helpful to order some samples of connectors to test fit. It's sometimes possible to partially plug in even a keyed connector the wrong way and cause a short or arc, or otherwise damage your device. Playing with samples will help you identify issues like that, which you wouldn't find in the datasheet. Connectors that have a physical clicking or snapping sensation when successfully mated should be preferred, since that will give you and your assemblers positive affirmation that the connectors are actually mated. The connector industry calls this *connector position assurance (CPA)*.

It's always a good idea to buy both sides of a connector pair from the same manufacturer so they mate with the right tolerances. If, for example, you buy an MCX male connector from one company and an MCX female connector from a different company, they'll mate, but the connection won't be as solid as if you bought both sides from the same manufacturer. The connection will still work, but since the tolerance stackup of a connector is designed with a particular mate in mind (namely the corresponding one made by the same manufacturer), you're more likely to have intermittent failures or weak connections. Mixing manufacturers will likely result in a connection that doesn't meet all of the claimed specifications in either datasheet. Even if you buy both connectors from the same manufacturer, it's still a good idea to get samples and test fit them. Connector companies sometimes acquire each other, which can mean that connectors from different acquisitions may not work well together. Some variations of connectors within a family also may not work well.

Interference and Noise

Cables are one of the most common reasons that products fail EMI testing. This is because cables can act like antennas and conduct noise into your product or radiate noise out of your product. To mitigate this, you need to figure out whether your cables will be considered "distributed" length or "lumped" length. A cable that can be considered distributed is electrically long, meaning it should be treated like a transmission line, not just a wire. A cable that can be considered lumped can be thought of as just a wire.

To figure out if a cable is lumped or distributed, we need to consider the rise time of the signal going through it. We also need to take into account the material that the conductor in the cable is surrounded by. In the case of a regular wire, it's usually a material like PVC, and in the case of coaxial cables, it's often a kind of Teflon. Whatever the material, you'll need to figure out the relative permittivity, which you can look up online.

The easiest way to explain how to classify a cable as lumped or distributed is with an example. Let's say we have 5 MHz square wave going down a cable. We look at the 5 MHz square wave on an oscilloscope and see that it has a rise time of 1,000 picoseconds (ps). The wire has an insulating jacket made of PVC, so we Google the relative permittivity of PVC and discover that it's about 3.0. First, let's calculate the delay of the signal through this material using this equation:

$$\text{delay} = \frac{\sqrt{\epsilon_r}}{c}$$

Here, ε_r is relative permittivity and c is the speed of light in a vacuum, which is a constant. To make the units easy for this method of calculation, c needs to be in inches per picosecond, which is 0.0118. We plug in those numbers like this:

$$\text{delay} = \frac{\sqrt{3}}{0.0118} = 146.78\,\text{ps} / \text{in}$$

Now we can calculate the electrical length of the rising edge as follows:

$$\text{rising edge electrical length} = \frac{\text{rise time}}{\text{delay}}$$

Because of the units we used for the speed of light, the rise time needs to be in picoseconds, and delay is already in picoseconds per inch. Let's plug in our numbers:

$$\text{rising edge electrical length} = \frac{1{,}000}{146.78} = 6.8\ \text{inches}$$

If the physical wire length is shorter than one-tenth the rising edge electrical length, it's lumped; if it's longer, it's distributed. In our example, if the wire is shorter than 0.68 inches, we can treat it as lumped. If the wire is longer than 0.68 inches, we need to treat it as distributed.

What do you do with that information? If a wire should be treated as distributed, what you really have is a transmission line. To minimize interference and noise, you should twist wires together. If you have a differential pair, the signal wire and its conjugate wire should be twisted together to reduce the electromagnetic emissions caused by the signals and reduce the susceptibility of those signals to external noise. Twisted pairs also help prevent crosstalk from other signals sharing the same cable.

If you have multiple twisted pairs in the same cable assembly, use different twist rates on each pair. If two twisted pairs have the same twist rate and lie next to one another, the benefits are somewhat degraded; each conductor is spending the same amount of time next to its corresponding conductor in the other twisted pair, so some crosstalk can still occur. You can buy assembled cables that do this for you, the most common being Ethernet. If you have a single-ended signal on a cable that should be treated

as distributed, twist it together with its reference, which in most cases is ground.

Another way to reduce external noise and susceptibility in a cable is to use a shielded cable. It's best to use a preassembled shielded cable rather than build one. The shield needs to be grounded at both ends to be effective for an electric field, but it only needs to be grounded at one end to be effective for a magnetic field. Be aware that many cables that claim or appear to be shielded are actually not well shielded. The only way for a shield to be effective is to have a complete seal around the entire perimeter of the connector housing—a technique called 360-degree shielding. Many digital cables like HDMI and USB are often not built correctly. (For more information about reducing EMI/EMC in cables, see Chapter 8.)

RF Connectors and Cables

Avoid building your own RF cables if at all possible, especially for high-frequency designs. It's easy to build a poor-quality cable if you don't have experience, and there's nothing more frustrating than debugging an insidious problem that ends up being a bad cable (trust me).

If you need an impedance-controlled cable, use coaxial cable, commonly known as *coax*. There are lots of different kinds of coax with different losses, impedances, and dielectric materials. The most common coax cable to use for 50 Ω applications is RG-58/U, and the most common for 75 Ω applications is RG-59/U. Make sure you pick a cable type that can handle the frequency, power, and impedance of your application.

Certain types of coax use braided outer conductors, which can cause signal to leak out. This can be a problem if you have multiple coax cables bundled together. As an alternative, look for cables that use a foil over the braid or a solid outer conductor. RG-223, rigid, and semi-rigid coax are all good options, but be aware that rigid and semi-rigid cables will be significantly less flexible and more expensive.

Not all connectors can handle all frequencies. For most RF applications, SMA connectors are fine. However, if you're doing work above about 18 GHz, you'll need more exotic (and unfortunately, more expensive) connectors. Table 3-5 lists many different RF connectors and their maximum usable frequency. Use this table only as a reference. Check the

datasheet for the specific connector you use to get the most accurate maximum frequency.

Table 3-5: RF Connectors and Their Maximum Frequencies

Connector name	Max. frequency	Connector name	Max. frequency
Twinax	200 MHz	QN-Type	11 GHz
UHF	300 MHz	N-Type	11 GHz
Triax	300 MHz	C-Type	11 GHz
MHV	500 MHz	MHF 4L LK	12 GHz
Mini-BNC	1 GHz	MHF 4L	12 GHz
F-Type	1 GHz	MHF 5	12 GHz
Mini-SMB	2 GHz	SSMA	12.4 GHz
Mini-UHF	2.5 GHz	MHF 5L	15 GHz
G-Type	3 GHz	MHF 7S	15 GHz
L	3 GHz	SMA	18 GHz
SMZ	4 GHz	APC-7 / 7 mm	18 GHz
SMB	4 GHz	Precision N	18 GHz
HN-Type	4 GHz	Precision TNC	18 GHz
FAKRA	4 GHz	U.FL / UMCC	18 GHz
BNC	4-12 GHz	OSP	22 GHz
QMA	6 GHz	GMS	26.5 GHz
MMCX	6 GHz	OSSP	28 GHz
MCX/OSX/PCX	6 GHz	3.5 mm	34 GHz
HD-BNC	6 GHz	SSMA	38 GHz
AMMC	6 GHz	SMP	40 GHz
AMC	6 GHz	2.92 mm	40 GHz

Connector name	Max. frequency	Connector name	Max. frequency
AFI	6 GHz	GPO/OSMP/SMP	40 GHz
OSMT	6 GHz	K-Type	40 GHz
MHF-TI	6 GHz	OS-50P	40 GHz
7/16	7.5 GHz	Precision (APC)	50 GHz
MHF I	9 GHz	2.4 mm	50 GHz
MHF I LK	9 GHz	V-Type	65 GHz
MHF 4	9 GHz	1.85 mm	65 GHz
SSMC	10 GHz	SMPS	65 GHz
SMC	7–10 GHz	1.0 mm launch	110 GHz
1.0/2.3	10 GHz	0.9 mm	120 GHz
TNC	11 GHz	0.8 mm	145 GHz
SC-Type	11 GHz		

When selecting RF connectors, prefer connectors that screw together rather than those that snap together. They're more reliable, and you can use a torque wrench to ensure that they're firmly attached (but not over-tightened). When you're screwing in connectors, always rotate only the nut instead of twisting the mating connector or cable.

Be sure to check the price of both sides of the RF connector. For example, an MCX female connector may only cost a couple of dollars, but surface-mount MCX male connectors are very hard to find and are about 10 times as expensive. That's because most applications use a female connector on both sides and a cable with a male connector on each end.

Not all cables and connectors can handle all environments. For example, if you need an RF connector that will work outdoors, an N-type connector is a better choice than an SMA connector. Think about where the cable will be sitting (inside an enclosure, buried under the ground, underwater, and so on) and the environmental stresses it will undergo (heating from the sun, rain, ice, and the like). Using a cable not rated for the environment in which it will be used can cause signal integrity problems or a catastrophic failure of

the cable, resulting in short or open circuits. Environmentally rated cables will often be thicker and less flexible than other cables.

Safety Considerations

High-voltage and high-current cables and connectors need to be safe. There are several standards and certifications that cables and connectors can be compliant with to assure safety. The difference between these certified parts and standard parts is mostly in the amount and type of insulation used. Don't pick cables and connectors so that they fit in your enclosure. Instead, pick cables and connectors that are rated for the current and voltage you'll be putting through them, and *then* design them into the enclosure.

All connectors and interconnects should make the connection with ground first and break connection with ground last. This is important for safety, especially when working with high voltages. Most connector manufacturers offer products that fit this description. They do this by making the ground conductor longer than the other conductors so that it touches the mating connector first when inserting the connector and is the last conductor to be pulled out of the mating connector.

Another safety feature that should be considered is exposed versus recessed pins and contacts. A high-voltage or high-current connection shouldn't have exposed pins that can be touched by the user or accidentally shorted against anything (necklaces, rings, keys, coins, and the like). Most connectors that are meant to be used at high voltages will take care of this for you, especially if they're UL or IEC certified.

Consider adding a strain relief feature to your enclosure if a cable will be external to the device. It will inevitably be yanked by the user, which can accelerate wear around the crimps on connector pins or even cause wires to fatigue and break. Whether you need to add strain relief will depend on the physical width and length of the cable, its placement, the gauge of wire, the cable assembly, and so on.

Naming Confusion

Beware of connectors that have similar names but are different. For example, MCX and MMCX are different and won't mate together. MMCX is smaller but looks otherwise identical to MCX. It's easy to order the wrong part or grab the wrong one if there's a pile of them mixed together.

Another classic example is SMA. SMA connectors are ubiquitous in RF engineering, but they're not the same as RP-SMA. The RP stands for *reverse polarity*. Confusing SMA and RP-SMA is even more insidious than MCX/MMCX because they will easily screw together and look connected but aren't. The reverse polarity aspect of RP-SMA means that screwing it into an SMA connector will result in either two receptacles next to each other or two pins next to each other. In both cases, the connection looks like an open circuit. In RF applications, that will cause a large reflection and a large voltage standing wave ratio (VSWR), and it can destroy components. RP-SMA is most commonly used in Wi-Fi equipment like routers, so if you unscrew the antenna from the back of your wireless router and screw it into your spectrum analyzer, the spectrum analyzer won't actually be connected to the antenna. It's a good idea to get a couple of RP-SMA to SMA adapters (you need both versions: hole-to-hole and pin-to-pin). They're cheap and can save you in a pinch.

One more set of connectors easy to get mixed up is U.FL connectors. U.FL is a trademarked name by Hirose, the manufacturer that invented it. Hirose has several size variants of U.FL, including E.FL and W.FL. Because other companies legally can't call their connectors U.FL, they have to call them something different. Other names used include IPEX, IPAX, IPX, AMC, MHF, and UMCC. All of these will mate with a U.FL connector, but there are lots of other tiny RF connectors that look very similar and do *not* mate with U.FL. They're all very small, look about the same, and you'll only realize they won't fit when you try to actually plug them in. To help avoid this problem, buy both mating sides of the connection from the same manufacturer. Connectors bought from one manufacturer will usually list mating connectors in the datasheet, removing the risk of a connection that doesn't quite fit right.

Crimped Connectors

Many people like to avoid crimped connectors because you have to buy the parts and a crimping tool and then take the time to build the cables (which, if done incorrectly, results in a fragile cable). They'll say it's much faster and easier to just solder wires without using any connectors. But in some cases, using crimped connectors can actually make things easier in the long run, especially in the prototype stage.

The advantage of crimped connectors over soldered connectors or even just soldering wires straight to pads is mechanical reliability. When you solder stranded-core wire, the solder will wick up the wire underneath the insulation. This creates a weak point where the now rigid, solder-containing part of the wire meets the wire without any solder and causes the strands to break under repeated strain or movement. This effect, which also occurs to a lesser degree with solid-core wire, can be enormously frustrating, especially when the wire breaks underneath the insulation, making the break difficult to find.

A crimped connection gets around this problem by using no solder and only the pressure of the bent metal fingers of the crimped terminal to make contact with the wire. Correctly executed, a crimped connection will last much longer than a soldered connection, and it's more robust to bending, plugging and unplugging, and strain. Ultimately, while soldering a wire to a pad may initially save you a little time building your prototype, using a crimped connector will help make your prototype more robust and make it easier to take apart, repair, and reassemble things without having to use a soldering iron every time.

Choosing Crimping Pins

Crimping pins are available with several different types of surface plating. The most common plating options you'll see are tin, gold, and palladium-nickel. Gold-plated or palladium-nickel-plated contacts provide more reliable and long-lasting connections. If you can't use those because of cost, you can use tin-plated contacts, but you need to perform thermal and vibration testing on them first. Tin-plated contacts can corrode, have a higher resistance, and can cause tin whiskers (more on that in Chapter 7).

WARNING

Never mate a gold contact to a tin contact. This is known to cause corrosion and reliability problems. Mate only gold to gold, tin to tin, and palladium-nickel to palladium-nickel.

If you can spend the money, buying premade cables will save you time and guarantee reliability. If you decide to build your own cables, use crimping pins and housings that have been sourced from a legitimate

manufacturer. The risk with buying off-brand crimp connector parts from different sources is that they are not always compatible with each other. If you buy a housing from Molex but the crimp pins from a Chinese eBay supplier, there's a good chance they're not going to fit together very well, or even at all.

Choosing Crimping Tools

Building cables with crimped connectors requires a crimping tool. For every family of crimped connector, there's a tool made by the manufacturer that's designed to perfectly crimp the pins. Unfortunately, a classic "gotcha" that connector manufacturers like to play is to make the connector housing and pins cheap but the crimping tool very expensive. For this reason, hobbyists and budget-constrained startups should always check the price of the crimping tool as well as the connector parts if you're building your own cables. Sometimes, the official tool is reasonably priced, but other times it can cost over $1,000.

To save money, you can try buying cheaper generic versions of crimping tools, but they don't always work well. When possible, and especially when you're working on a safety- or reliability-critical design, get the official manufacturer tool; it will produce much more consistent and reliable crimps. If you can't afford the official tool new, try checking eBay for used ones. You can sometimes find them significantly cheaper there, but they're not always available. Another place to check is *https://www.crimptools.com*, which sells lots of used crimpers. Remember that official crimping tools typically only work with a single connector family, so if you want to use two different connector types, you'll need two different crimping tools.

If you want to save money by using a generic or knock-off crimping tool, make sure to carefully follow the crimping instructions provided by the connector manufacturer. These instructions will include detailed measurements and drawings showing what a perfectly crimped connection looks like, which will help you identify and avoid issues with a generic crimping tool. Another great resource is the Molex *Quality Crimping Handbook*, available at *https://www.molex.com/*. This handbook contains detailed information on correct crimping technique, how to test and troubleshoot crimps, and lots of other very practical tips that apply to all connector families, not just Molex.

Table 3-6 compares the cost and efficacy of different crimping tools for common connectors. This is certainly not an exhaustive list, but it's a good place to start.

Table 3-6: Crimping Tools for Common Connectors

Connector	Pitch	Crimping tool	Cost (in 2025)	Tool origin	Recommended?
Mini-PV, DuPont	2.54 mm	HT-95 / HT-0095	$1,600	Manufacturer	Yes
		Harwin Z20-320	$600	Generic	Yes
		Hozan P-707	$50	Generic	Yes
		Engineer PA-09	$42	Generic	No
		Hozan P-707	$50	Generic	Yes
		PR-3254	$37	Generic	Yes
		IWISS SN-025	$23	Generic	Yes
JST, PH	2 mm	WC-240	$540	Manufacturer	Yes
		Engineer PA-09	$42	Generic	Yes
		IWISS SN-01BM	$23	Generic	Yes

Connector	Pitch	Crimping tool	Cost (in 2025)	Tool origin	Recommended?
JST, XH	2.5 mm	WC-110	$540	Manufacturer	Yes
		Engineer PA-09	$42	Generic	No
		IWISS SN-01B	$23	Generic	Yes
Picoblade	1.25 mm	Molex 63827-1400	$540	Manufacturer	Yes
		Molex 63827-1500	$700	Manufacturer	Yes
KK 254 / KK 100	2.54 mm	Molex 63811-8200	$430	Manufacturer	Yes
		IWISS SN-01BM	$23	Generic	No
SL	2.54 mm	Molex 63825-8800	$450	Manufacturer	Yes
		Molex 63811-8700	$470	Manufacturer	Yes
		Molex 64016-0201	$174	Manufacturer	Yes
		IWISS SN-2549	$23	Generic	Yes

You'll notice that there are several generic crimpers that work for multiple types of connectors, but they don't always work equally well on all of them. If you search Amazon or eBay for crimpers, you'll likely find several offerings not listed here that look similar but are made by different Chinese companies. These brands come and go, and it's hard to know whether they'll work well. It may be worth it to take the risk and bet your $20 that you can get a good enough result if spending $500 on the official tool isn't an option. Also be aware that sometimes various wire gauges require different tools, even if they're in the same connector family, so before you buy a crimping tool, check to make sure it can crimp the wire gauge you want.

I also recommend checking out Matt Millman's website, *https://www.mattmillman.com/info/crimpconnectors/*, which has a wealth of information about a wide range of connectors and crimping tools.

Circuit Protection

Circuit protection isn't just important for passing certification tests. It's also an important part of building reliable hardware that can survive the harsh testing conditions and mistakes that may come up during the PCB bring-up process. There are lots of different things to protect your designs from, so it's useful to think in terms of building up layers of defense. Here are some ways to protect against the most common problems your design may encounter.

Filters for Electromagnetic Interference

Filters can be an important part of your EMC strategy. However, it's important to know that even if you choose the right filter, you can still render it useless if you integrate it into your design incorrectly. It's also a good idea to add EMI filters only if you're confident that they're necessary (determined by either calculation or measurement) or if the engineer at your testing facility recommends one. Otherwise, you'll just be adding unnecessary cost to your BOM. Don't finish reading this section and start adding EMI filters everywhere unless you know you need to.

You have two choices when selecting an EMI filter: single-pole or multipole. The most common single-pole filter for EMI suppression is a feed-through capacitor, which are most often used at transitions in the

layout. If you have a critical, high-speed signal leaving a shielded enclosure, a feed-through capacitor will pass the signal through the shielding and provide capacitance with very low inductance to ground. This ensures that high-frequency noise can actually be bypassed to ground without seeing impedance from stray lead inductance. Feedthrough capacitors are typically bulkhead-mounted parts that screw into a hole on the side of an enclosure or shield. Some connectors have built-in feedthrough capacitors that make them easier to integrate.

Another form of feedthrough capacitor is a three-terminal capacitor. These are usually surface-mount devices that have, as the name implies, three terminals and either three or four pads. The first two terminals are for the signal input and output, and any remaining terminals are connected to ground. Again, this ensures extremely low parasitic inductance on the capacitor so high-frequency noise can be bypassed. Three-terminal capacitors are useful when crossing shielding boundaries on a PCB.

Multipole filters can be bought in a single package, or you can design them yourself. Figure 3-10 shows the four kinds of multipole filters you'll encounter.

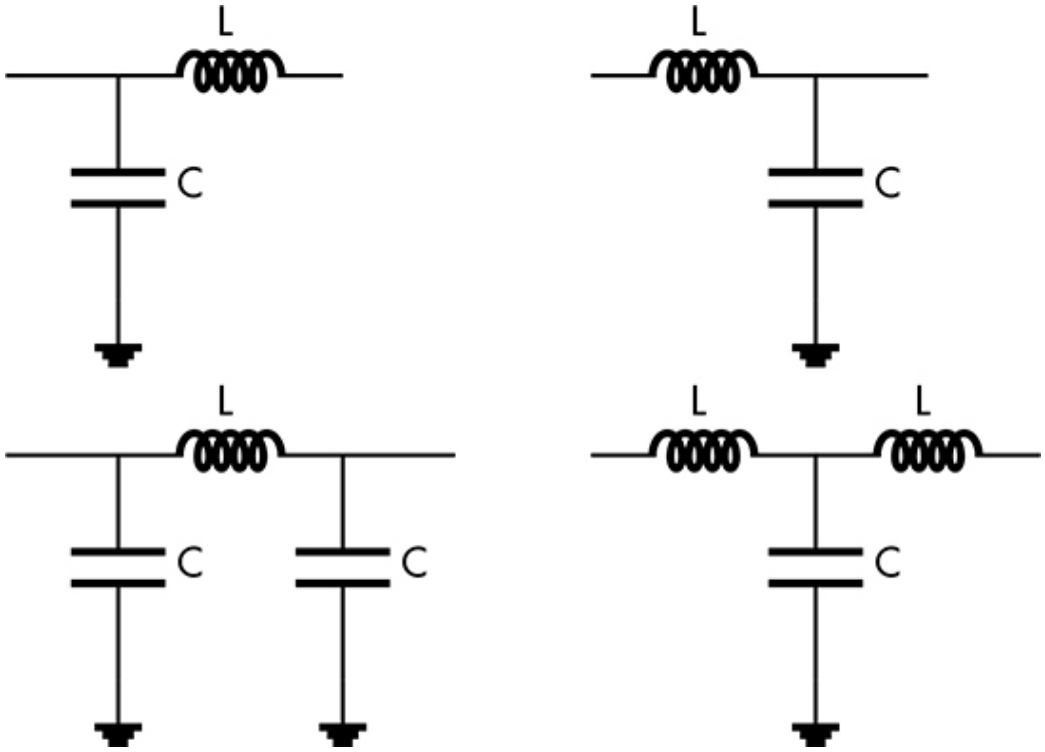


Figure 3-10: A shunt-series filter (top left), series-shunt filter (top right), Pi filter (bottom left), and T filter (bottom right)

All of these topologies are low-pass filters, since we're trying to filter out high-frequency noise. To determine which topology you should use, first determine the impedance you need on the input and the output of the filter. A shunt-series filter has a high-impedance input and a low-impedance output, while a series-shunt filter has a low-impedance input and a high-impedance output. A rule of thumb is if the input source has an impedance of less than 100 Ω, use a filter with an inductor as the first element. If the input source has an impedance of greater than 100 Ω, use a filter with a capacitor as the first element.

All EMI filters have an insertion loss. This gives you a benchmark for comparing EMI filters against each other and choosing one appropriate to your application. You want high insertion loss at the frequencies you're trying to filter out and low insertion loss at the frequencies you want to keep. As with other passive components, you'll want to look at the plot of filter performance over frequency in the datasheet before selecting a part. Like ferrite beads, EMI filters should have at least a 5x frequency passband. In other words, if you're filtering a 5 MHz signal, your filter shouldn't have significant insertion loss until 25 MHz. You want to remove high-frequency content and slow down the signal rise time but still preserve enough harmonics of the signal to prevent significant distortion.

All ESD or EMI suppression devices should always be connected to chassis ground rather than the ground plane of the PCB. This is very important to ensuring that the devices actually work as intended. To be able to do this, your chassis must be made of metal. Many consumer products have a plastic enclosure, but this makes it much more difficult to control EMI and EMC. You should prefer a metal enclosure if possible, and in some products with more stringent EMI requirements (like medical devices), it's extremely difficult to make a product that will pass those regulatory requirements without a metal enclosure. Note that while ESD and EMI devices should be connected to chassis ground, capacitors in low-pass filters used for EMI control should be connected to the PCB reference plane and not to chassis ground in order to keep stray inductance to a minimum.

Overvoltage Events

Overvoltage events are something else you need to protect against. They're often the result of ESD, which occurs when any object that has been raised to a much higher potential than the PCB makes contact with the PCB. Your

users will be a common source of ESD, as simply walking around on a carpeted floor on a dry day can build up a massive potential that's equalized when they touch your product. That means your device needs to be able to survive a very fast, very high voltage spike. For example, a typical voltage transient might be 5 nanoseconds (ns) long and 10,000 V. A solution is to use ESD diodes.

ESD diodes are specially designed diodes that have a very fast response time and can shunt the high voltage (and the resulting current) from an ESD event to ground. A lot of ICs have ESD protection already designed into them, but some standards require a more substantial amount of ESD protection (given as the maximum voltage spike that the device can tolerate from an ESD event). As such, even though your ICs may have protection already built in, you may need to add additional ESD diodes to cover the full range required. For example, many ICs only provide up to 5 kV of protection, but some medical standards require 15 kV of protection.

When deciding where ESD diodes are necessary, think about where the device can be touched. Consider not only directly accessible conductive surfaces but also conductive surfaces that are close enough to the exterior of the device that they can be arced to. A 10,000 V pulse can easily arc to a recessed signal pin on a connector, even if it's not possible to touch the pin directly. If you need to have ESD diodes on high-speed signals or on sensitive analog signal lines (for example, on a medical device that will make contact with a patient), use diodes with as low capacitance as possible. This will prevent excessive capacitive loading, and it will help prevent noise from capacitively coupling into your sensitive signal. Polymer ESD devices are a good option when low capacitance is required. These devices typically aren't actually diodes but rather use something called a variable voltage material.

Another ESD mitigation approach is to use *steering diodes*. This refers not to a particular kind of diode but a particular diode circuit. A combination of ESD and transient voltage suppressor (TVS) or Zener diodes are used together to divert voltage transients away from sensitive components. You can buy these circuits all in one IC as "steering diode arrays."

If you need to design to a particular standard, make sure you pay close attention to the exact ESD requirements, not just the maximum voltage you must sustain. There are different models of ESD discharge that are commonly used, like the human body model, the machine model, and the

charged device model. These are all tested in different ways, and they all have different output currents and peak times. Carefully read all of the ESD requirements of the standard you're following, and don't be afraid to get expert advice from an organization like UL to help understand what applies to you. If you're not required to follow a particular standard but still need ESD protection, reading through standards like IEC 61000-4-2 can help you figure out what protection might be appropriate for your device.

While voltage spikes often result from ESD, they can also come from more mundane sources. Whenever an inductor is switched on or off, for example, a voltage spike will occur because the inductor will create a voltage that opposes the change in current. This voltage spike stems from the following relationship:

$$V = -L\frac{di}{dt}$$

This applies not just to discrete inductors in your circuit but also to things that act like inductors, including wires and even motors. For example, if you're switching a motor on and off quickly or changing directions rapidly, its windings will act like an inductor, and you'll see a voltage spike whenever the current through the motor changes. This spike is often clamped with a diode (referred to as a *flyback diode* when used for this purpose). The same effect can be seen when switching a transformer.

One of the most common methods for preventing overvoltage conditions is to use *TVS diodes*. Any voltage above some device-specific threshold will be clamped down to a maximum of that threshold. TVS diodes have a very long lifetime. They can also clamp very quickly, as fast as 50 ps, and TVS diodes in SMT packages can have particularly fast response times, since they can have a low lead inductance. Normal TVS diodes can protect from voltages as high as several hundred volts, but TVS diodes meant for ESD protection can be rated as high as 15 kV. TVS diodes are usually physically small and therefore can't dissipate a large amount of energy. That means that any voltage spikes that last longer than a few microseconds can destroy the part.

Spark gaps are often used to shunt very high voltages too. A spark gap can be implemented on a PCB by pulling back solder mask around a small area of ground plane. You can see an example of this in Figure 3-11. The trace running across the top of the board is ground, and there are tiny

rectangles that have been exposed directly across from each through-hole connector pin.

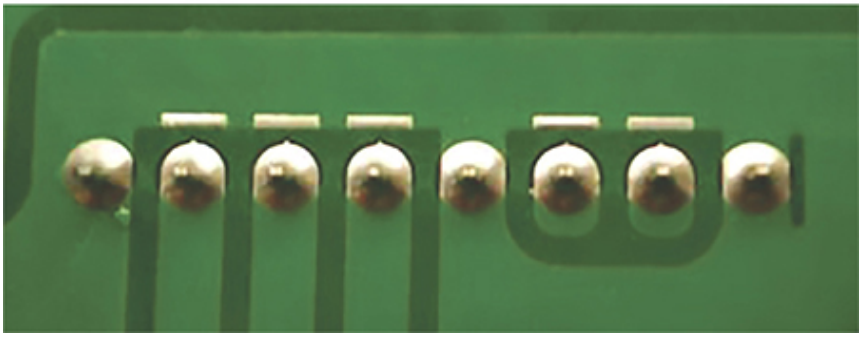


Figure 3-11: A spark gap on a connector

You shouldn't rely upon spark gaps as the only method of reducing voltage spikes, however. Their performance varies greatly based on the condition of the PCB, the humidity, and how ablated the spark gap is as a result of previous sparks. Spark gaps will also only reduce the voltage to a few hundred volts, which is still enough to damage a PCB. Where a spark gap is most useful is in preventing arcing across other protective components. For more consistent performance, consider using a *gas discharge tube (GDT)*. These components are designed to handle large amounts of energy, and despite a name that makes them sound huge, they're available in relatively small through-hole and surface-mount packages.

Overvoltage events from lightning strikes require special consideration, with protection against lightning strikes falling under two categories: direct and indirect. The only way to survive a direct lightning strike is to divert the strike with a lightning rod and prevent it from getting to the PCB in the first place. Indirect lightning strikes are large surges that originally came from lightning striking another object. The methods used to suppress indirect lightning strikes are the same used for protecting against any extremely large voltage spikes. Chassis ground should be connected directly to earth ground. Connect all grounds together at one point with a 1 MΩ resistor and a 2 kV 0.1 µF capacitor in parallel. The capacitor will provide a path to ground for fast transients, and the resistor will prevent charge from accumulating on the PCB.

Silicon-Controlled Rectifier Latch-Up

Silicon-controlled rectifier (SCR) latch-up is another danger to watch out for. An SCR is a semiconductor made by alternating P-N-P-N doped silicon. It has three terminals and doesn't start conducting until the third terminal reaches a voltage above a particular threshold. After that, the SCR will conduct even after that triggering voltage is removed. SCR latch-up happens when you accidentally build an SCR in your IC and it gets triggered. Any place where the P-N-P-N structure is found, even if it's not designed as an SCR, will have this effect. It doesn't happen all the time since you typically need a very large voltage to trigger the effect, but because that triggering voltage doesn't need to remain there after the SCR is conducting, a transient voltage spike can sometimes be enough to induce latch-up.

Latch-up is most commonly seen when there's a large positive or negative voltage spike on the pin of a digital IC or when applying a supply voltage above the rated absolute maximum. One example where this might happen is if you have multiple power domains and each power supply takes a different amount of time to turn on. If the outputs aren't sequenced correctly, an unpowered IC might get a voltage on an input pin that causes latch-up on that chip. There's more information in Chapter 5 on how to prevent these kinds of sequencing errors.

ESD events can also cause latch-up. If you work in the aerospace industry, you'll be familiar with single-event upsets (SEUs) caused by high-energy particles colliding with your electronics. In space, high-energy cosmic radiation can collide with the sensitive electronics on a satellite or other spacecraft and cause latch-ups. A total power cycle is the only way to clear a latch-up, so satellite power circuits often have latch-up detectors and can automatically reset themselves.

Radiation in space is one of the reasons why spacecraft (and vehicles that operate at very high altitudes or are otherwise subjected to high radiation) often use *radiation-hardened (rad-hard)* chips. However, rad-hard chips are *very* expensive and are usually several generations behind the state of the art in consumer electronics. This is simply because demand isn't that high, and engineers prefer chips that have been proven over time to be reliable. There's some additional benefit from the physically larger process sizes that older chips use, since they're a little less susceptible to the effects of radiation.

Another effect of SEUs is bit flipping. A high-energy particle striking memory can flip the stored bits. Some designers address bit flips without

using rad-hard parts by using distributed consumer-grade parts designed in a failure-tolerant architecture. Multiple chips running the same code in parallel are physically spread out over a spacecraft (to try to spread out the damage a burst of radiation will have) and vote on the correct output given the same inputs. If three out of four chips agree that something should happen, then it's likely that the fourth chip has been damaged in some way. A governor circuit can then trigger a hard reset of the suspected bad chip and hopefully fix the problem. If you're building something that will go into space and decide to go the cheaper consumer route (a lot of companies are doing this now), check out the MSP430 from TI as a microcontroller. Specifically, the MSP430s that use FRAM have been found to be highly resistant to radiation. They aren't sold as rad-hard, so there are no guarantees, but that also means they're extremely cheap compared to rad-hard microcontrollers.

Fuses for Overcurrent Events

If there's a danger of an overcurrent event, use a fuse. There are two main kinds of fuses: fast-blow and slow-blow. A fast-blow fuse is designed to blow (become an open circuit) immediately after a certain current threshold is exceeded. A slow-blow fuse has a time delay; it will blow after the current threshold is exceeded for a certain amount of time. This allows fast-current transients through but prevents a continuous overcurrent condition like a dead short.

To pick a fast-blow fuse, use this equation:

$$\text{fuse current rating} = \frac{\text{nominal operation current}}{\text{temperature derating} \times 0.75}$$

If you're operating the fuse at room temperature, use a temperature derating factor of 1. As you increase the operating temperature of the fuse, though, you need to start accounting for the fact that the fuse will blow at a lower current. You can find the temperature derating factor in the fuse's datasheet.

Choosing a slow-blow fuse isn't quite as simple as selecting a fast-blow fuse. Like ferrite beads, you'll need to look at the plots in the datasheet to figure out what will prevent damage to your device. In general, you want fuses to have a low DC resistance so they'll have a low voltage drop. No matter what fuse you pick, the only way to be absolutely sure that it will

blow only when you want is to test it yourself in conditions as close as possible to the real application. Different sources will tell you to derate different amounts, and even different manufacturers have different methods and rules for rating their components. Choose parts based on the graphs in the datasheet, and then verify that they'll work for you by testing.

You'll sometimes see fuse datasheets mention I^2t, or melting point. The higher the I^2t, the longer the fuse will take to blow. A fuse may experience pulses of current that don't reach the melting point but still cause fatigue in the fuse. Datasheets sometimes list the "pulse withstanding capabilities" of a fuse, which describes how many pulses of a given percentage of the I^2t it can survive. Another important specification is the breaking capacity or interrupt rating. This is the point where a fuse could catastrophically fail, so make sure you never exceed it.

The problem with fuses is that once they blow, they need to be replaced. (Of course, blown fuses should be replaced after the problem that caused the fuse to blow in the first place has been identified and fixed. Don't just replace the fuse and expect the problem to go away, and *definitely* don't short out the fuse to get it to quit blowing!) Fuses are available in soldered packages that require desoldering to replace, or they can be used with fuse holders that make testing and replacing them much easier. The main deciding factors on whether you can use a fuse holder are the amount of space you have (since fuse holders take up extra height), the size of the fuse you need (fuse holders aren't available for every single fuse), and your budget (fuse holders are another BOM item that add cost). If these constraints allow it, use fuse holders; they make repair much easier, even if they aren't user accessible. However, you should consider the clearances and steps required to access the fuse for replacement during your enclosure design. You can find both SMT and through-hole fuse holders, and the SMT variants get surprisingly small. This lets you fit fuses in smaller enclosures, but of course it also means that the fuses won't be able to handle as much current as physically larger fuses.

To avoid the problem of replacing blown fuses, you can use a *positive temperature coefficient (PTC) resettable fuse*. This is a small passive device that sits in series with your power input. As current increases through one of these devices, the resistance also increases. That's why these components are referred to as resettable fuses: They act like an open during high currents but look like a short once the current returns to normal levels. That means

you don't need to replace them every time an overcurrent event happens. The downside to PTCs is that they can drop significant voltage and can have a high ESR. They're also extremely temperature sensitive. On the other hand, they're very low cost.

Fuses work well for very large voltages and currents, and PTCs fix the problem of having to replace a part whenever it fails, but there's an option that performs even better than a PTC: Several manufacturers make ICs that are specifically designed to prevent overcurrent and overvoltage conditions. They don't have a standardized name, but you can search for things like *overvoltage clamp*, *surge suppression IC*, or *overcurrent protection controller* to find them. They all work in different ways, like a sense resistor and a FET or a modified crowbar circuit, for example. These chips will behave similarly to a PTC, but without the large voltage drop and sometimes with a faster response time. Some ICs will also give you digital outputs that you can use as an input to a microcontroller to detect faults and react to them. Other ICs have a programmable current limit. It's hard to recommend a single chip since there are so many and they all do different things, but take the time to shop around and see what's available. It can save you a lot of complexity in your schematic while also affording a great deal of robustness.

Reverse Bias Protection

There are ICs that will provide reverse bias protection. That way, if someone manages to defeat any mechanical barriers to reverse biasing the device (for example, by forcing batteries in backward or plugging in an unkeyed power cable the wrong way), your device won't be damaged. These ICs sometimes involve an ideal diode circuit, which uses a FET to look like a diode with a forward voltage of 0 V. The ideal diode circuit can be internal to the IC or external, but the application note will tell you exactly how to implement it.

For a product that runs on user-installed batteries, reverse bias protection is essential. Many products use mechanical means to prevent reverse bias by molding the battery compartment such that only the positive bump on the battery can make contact with the positive terminal of the device. This method can be almost free to implement and works well. Reverse bias protection can also be very useful on PCBs that are being tested or prototyped in the lab, since it's easy to get the test leads coming from a power supply into your test board mixed up and reverse-bias it.

4
SELECTING ACTIVE COMPONENTS

Generally, active components will define the major capabilities and specifications of your designs. You'll spend most of your component-selection time looking for your active components, so it's important to look for the right specification. In this chapter, we'll look at some of the most common categories of active components and discuss what to consider when selecting them.

Advertised vs. Actual Performance

As we explore active components, keep in mind that datasheets have been known to fib about the performance of some parts. After you gain some experience scanning distributor websites for particular parts, you'll get an idea of what's on the market. If you see a part that claims performance that seems too good to be true, it probably is. Datasheets love to use flashy claims on the first page to draw you in. For example, wideband amplifiers that claim 20 dB of gain will, upon closer inspection, actually only give you 20 dB of gain at exactly one input voltage, one input RF power, and one frequency.

For a more realistic view of a part's performance, test it on an evaluation board first, especially if you're planning to use the part at or

near any of its limits. You can sometimes get evaluation boards for free through sales engineers, but you may need to buy them. If they're too expensive, you can try fabricating one yourself, since evaluation board Gerber and manufacturing files are almost always released publicly or available upon request. The idea is that you can verify that the datasheet is telling the truth and that the part will perform the way you expect in your application before you commit to ordering it in large quantities.

Oscillators and Crystals

When oscillators are used as an external reference for an IC, the datasheet and application notes often give oscillator parameters needed to guarantee nominal IC operation. They'll sometimes also recommend specific part numbers. Follow these recommendations and use the suggested part if possible. That said, sometimes the suggested part numbers will be end of life or unavailable, so don't just blindly use the recommended parts without checking that they're still active and in stock with an authorized distributor.

Crystal oscillators typically need two external capacitors, called *shunt capacitors*, to function correctly. You can get the value of these capacitors from the datasheet or you can calculate it. A good reference for these calculations is the *Guidelines for Oscillator Design on STM8AF/AL/S and STM32 MCUs/MPUs* (application note number AN2867, available as a free PDF from ST Microelectronics). That design guide, while written for specific microcontrollers, contains a wealth of general information about oscillators and crystals, so rather than re-create it here, I recommend giving it a look. In most circumstances, the best thing to do is use the values specified by the oscillator's datasheet. Always use NP0/C0G-type capacitors. You can optionally decrease the capacitance by 1 or 2 pF, depending on the physical size of the capacitors you're using to compensate for the extra capacitance you'll get between the capacitor terminals and PCB's ground plane. As long as you're within a couple of pF of the correct value, the oscillator should behave just fine.

If you plan on using a crystal oscillator outside of its temperature range, you need to characterize its performance yourself. The datasheet

will only guarantee performance within a certain temperature range, and outside of that range, the crystal may experience an *activity dip*. This is the point where the crystal suddenly increases in resistance, which can cause a frequency error of up to 20 ppm that can make your system lose phase lock. These dips happen at very narrow temperature ranges, and they're almost never specified in the datasheet. If your design needs a highly accurate clock reference over a large temperature range, you should characterize the crystal you're planning to use and design around those points if necessary.

A crystal's resonant frequency also changes slightly over time as the crystal ages. Luckily, this change is very small, usually about 5 ppm during the first year and 10 to 20 ppm over the next 10 years. Still, for very sensitive designs, it's important to be aware of this fact. The aging effect isn't impacted by temperature.

Silicon oscillators offer several advantages over crystal oscillators. Instead of a piece of quartz vibrating, it's a MEMS structure on silicon. This allows silicon oscillators to be physically smaller than crystal oscillators and allows them to be programmable to a range of frequencies. Silicon oscillators can operate from 1 kHz up to 170 MHz and include EMI-reduction features.

Before 2018, silicon oscillators could fail if exposed to even small concentrations of helium gas. There was an incident where a bunch of technicians installing an MRI machine had their iPhone 8s all die because of a small helium leak. Eventually, the gas dissipated and the phones started working again. It turned out that helium molecules were small enough to migrate inside the silicon package of the oscillator and cause a significant frequency deviation. Since then, manufacturers have developed special packages and coatings that reduce or eliminate this problem. If you decide to use a silicon oscillator, you may want to find one that's rated to operate in atmospheres containing concentrations of small gas molecules, especially if you expect your product to have to operate under those conditions.

ADCS AND DACS

There's a huge amount of information to consider when choosing an ADC or DAC, and there simply isn't room in this book for the discussion they deserve. Instead, check out Tl's "Choose the Right A/D Converter for Your Application" and *The Data Conversion Handbook* by Analog Devices (2005). They're both among the resources linked on the website for this book.

Power Supplies

When choosing power supplies, take the time to calculate your power budget by determining the minimum and maximum current and power that each component in your design will require. Then make sure your power supplies can provide at least 10 percent more than what you need. Also make sure none of your components are dissipating more power than they're rated for, and that you've taken measures to handle any heat produced by those parts.

Voltage Conversion

A switched-mode power supply or DC/DC converter is the best way to go from a low voltage to a high voltage and vice versa. This is useful when you need to move large amounts of power. It's more efficient to send the power using a high voltage; the current will be lower, which will reduce your loss due to resistance in your traces, cables, or transmission lines. However, since most designs tend to run at a low voltage, you'll need to reduce the voltage back down again, which is where an SMPS comes in.

There are myriad SMPS and DC/DC converter ICs that you can use. TI has a nice, free online tool called WEBENCH (*https://webench.ti.com*) that will help you run some simulations and find a part that meets your requirements. SMPSs can be fickle if you don't use the right inductor, use a poor layout, don't have enough output capacitance, or run the part outside of its operating range. Make sure to use as many of the suggested parts in the application note as possible,

and follow the layout guidelines. There's more information about that in Chapter 6.

If the FET in an SMPS is switching below 20 kHz, you'll hear a soft humming or squealing noise coming from the inductor and capacitors. This is because some capacitors are slightly piezoelectric, and some wire-wound inductors will physically move slightly as their magnetic field changes, creating sound. You can solve this problem by running the SMPS above 20 kHz. This way, any sound produced will be above the human hearing range.

Sound shouldn't be the only reason you pick a particular switching frequency, though. More importantly, the switching frequency decides what value inductor you need, and the higher the inductance you need, the larger the physical inductor must be. Because there are so many different SMPS ICs, topologies, and applications, discussing exactly how to choose a switching frequency for your design is outside the scope of this book. The datasheet for your SMPS IC should have a section on switching frequency and usually some suggested inductors. The TI WEBENCH tool will also let you sort SMPS ICs by the physical layout area required for operation, which includes the inductor size. It's common for chips to abstract away the switching frequency design decision for you and simply call out a single inductor value to use to get the performance described in the datasheet. In any case, you need to be aware of the size of the inductor required for a particular IC, especially if you have a space-constrained design. Semiconductor companies can get tricky on you because they'll have big, glossy advertisements about how small their SMPS IC package is but conveniently leave out that the size of the required inductor is just as big or bigger.

SMPSs can cause your power lines to have extra voltage ripple caused by the inductor switching on and off. This ripple is characterized in the datasheet of the IC, but it's critical that you use the right capacitors to achieve datasheet performance. The number of capacitors, their type, where they're placed, and their value all determine the cleanliness of the power on the output of the SMPS. If ripple is important to you, check the part's datasheet before you choose to use it, and make sure it's acceptable even after derating it slightly.

Make sure you use DC/DC converters exactly as described in their data-sheets and application notes. If you're using a DC/DC converter in a non-standard way, talk to the application engineers or simulate it using a model from the manufacturer and real (non-ideal) components. LTspice is a good simulation tool for those kinds of situations, and it's free. Keep in mind that DC/DC converters are less stable at small (less than 20 percent) loads. Also, don't use the synchronization feature on DC/DC converters unless you really need it. It's easy to cause increased noise or oscillation if the feature isn't used exactly right.

Following the application notes for DC/DC converters will get you to a working circuit, but application notes don't always show the most optimal way to design with the part. They're usually pretty good, but, as an example, some DC/DC converter application notes will recommend placing a snub-ber circuit near the switching transistor to help reduce EMI. This will reduce EMI, but it will also reduce the DC/DC converter's efficiency. As a rule of thumb, follow the application note, but keep your wits about you and know that it's okay to change things if you know what you're doing.

Low-Dropout Regulators

If you use low-dropout regulators (LDOs), consider using parts rated for low noise. They usually don't affect cost and can have significantly lower ripple on the output. They can also save you a few components by reducing the number of output capacitors you may need.

Remember to compensate for the dropout voltage of your LDOs in your power budget. If you want 5 V on the output, you need to provide *more* than 5 V on the input. The lower the dropout of the part, the better, since less power will be lost as heat. LDOs operate most efficiently when their input voltage is *just* high enough to account for the drop, since any higher voltage will just be burned out as heat.

LDOs have a *power supply rejection ratio (PSRR)* that's given in the data-sheet. This is the attenuation in dB of the noise on the input of the LDO compared to the output. It's not uncommon for this value to be 40 to 50 dB, or even higher, depending on the dropout voltage. You can take advantage of this by using LDOs as a power supply filter. This can

be helpful for removing ripple from an SMPS or any noise picked up by a long power cable. A common technique is to use an SMPS to do large voltage conversions more efficiently and then put an LDO with a low dropout right before the ICs you're trying to power to help isolate them from the noise of the rest of the power nets.

Thermal Shutdown

One nice feature that many power supply ICs contain is thermal shutdown. An internal temperature sensor will automatically disable the regulated output once the chip gets too hot. Thermal shutdown is often triggered at very high temperatures, so reliability of the part will be poor if it's operated in a way that triggers thermal shutdown with any regularity. This feature is meant more as a safety measure.

Power supply ICs will sometimes also have a "power good" pin, which you can use to drive an LED or check with a microcontroller to detect fault conditions and take the correct action. Look for both of these features when shopping for power supply chips.

Microcontrollers

Before you go all in on a microcontroller, buy a development board for it. This will give you a good idea of how painful it is to actually get code running on the chip, and you'll get a feel for what development will be like. Development boards sold directly by the manufacturer are sometimes very expensive, but you can almost always find a cheaper third-party development board. Check sites like Tindie, eBay, or AliExpress. I also recommend talking to someone who has used the chip you're considering. They can be a great help in getting past the marketing veneer and finding out what it's really like to develop on the chip.

Before you decide on a microcontroller or processor, you should read its errata. This is a separate document you need to download from the manufacturer's website that lists all of the known problems or mistakes in the silicon, what their symptoms are, and what (if any) workarounds exist. Reading this document can save you days or weeks

of troubleshooting and, in extreme cases, can cause you to pick a different part.

Choosing the Right Microcontroller

Look for microcontrollers that already contain as many of your requirements as possible. For instance, if your requirements call for Ethernet, try to find a microcontroller with built-in Ethernet instead of having to use an external Ethernet IC. Avoid as much bit-banging as you can. A hardware implementation of a bus or protocol on your microcontroller will save you lots of development time, will generally have better performance, and is less likely to have bugs. On the other hand, bit-banging can be a great way to work around a buggy hardware peripheral.

Besides Ethernet, other common buses that some microcontrollers include are USB, I2C, SPI, USART, and UART. Different chips have different numbers of each of those buses, so if you have a UART-heavy application, for example, try to find a chip that includes multiple UARTs. You can also find microcontrollers that support specialized protocols like MIDI or CAN bus, and there are even microcontrollers with built-in radio modules. This can save you a lot on your BOM, since you can eliminate an entire external IC and just use your microcontroller. Sometimes microcontrollers won't let you use all peripherals all at the same time because of limitations in pin configurations, so you should download the manufacturer's configuration tool to verify that you can actually use the combination of peripherals you want.

When it comes to memory requirements, estimate the amount of both RAM and nonvolatile storage that you'll need. If you need more nonvolatile storage than your preferred microcontroller can provide, you can use an external storage component. SPI EEPROMs are common and can store huge amounts of information (this can be useful for storing lookup tables to improve application performance). Talk to your software team about how much RAM and storage they'll need to meet the design requirements. You'll also need them to weigh in on performance requirements: the number of cores and threads, clock

speeds, and whether any cryptographic or graphics acceleration cores are needed. If your software team is already familiar with a particular processor family or architecture, or has a lot of code and boilerplate written for it, you should probably give more weight to that processor family or architecture when selecting one.

Considering Software Development Environments

You should also investigate the software development environment required for a microcontroller before deciding on a part. If you decide to use the proprietary tools that the manufacturer provides, test them out first and take a look through the provided example code and libraries. You may be surprised at the poor quality of the code in the provided libraries, and it's likely that there will be bugs.

If you decide to use the manufacturer-provided IDE, make sure to budget for it. Sometimes they can be extremely expensive. Other times they cost money but have a free, feature-reduced version. Not realizing this beforehand could cause you to exceed your budget or waste time optimizing code size if you can't afford the IDE.

You also have the option of using a free, open source tool chain. There are lots of tools and support for almost every microcontroller out there, but because of that, you'll probably need to take some time to get your development environment set up for your particular chip. One IDE that does a good job of integrating a lot of tools together for you is PlatformIO (*http://platformio.org*). It doesn't have support for every single IC, but it does support a lot, especially for IoT applications.

Derating the Operating Parameters

It's always a good idea to derate microcontroller operating parameters so you aren't running them at their maximum ratings. Table 4-1 shows NASA's recommendations for derating integrated circuits.

Table 4-1: Integrated Circuit Derating

Stress parameter	Derating (digital)	Derating (linear)

Stress parameter	Derating (digital)	Derating (linear)
Max. supply/input voltage	90%	80%
Power dissipation	80%	75%
Max. junction temperature	80%	75%
Max. output current	80%	80%
Max. clock frequency	80%	80%

Keep in mind that these are guidelines, and since these guidelines are from NASA and intended for spacecraft design, they're conservative.

RF Components

There are a lot of nuances to consider when choosing RF components, and a full treatment of how to design an RF signal path is outside the scope of this book. In this section, we'll look at a few of the more universal RF part selection considerations that apply to complex designs as well as simpler and more commonly encountered RF circuits.

Frequency-Dependent Parameters

If you're picking out an RF part, pay attention to the insertion loss at the frequency you'll be using it. Insertion loss changes based on frequency, but the datasheet will usually advertise the minimum insertion loss across its entire frequency operating range. If you're not careful, your link budget will be wrong because you didn't account for the extra loss at your actual frequency of operation. As always, you need to check the plots in the datasheet and not go by the bullet points on the first page.

The higher the power that you're working with, the more important loss is in your parts. Consider a cable that you measure to have 1 dB of loss. At a low power, the difference of a single dB is a small amount of

absolute power. But at a high power, a single dB can be watts of difference. This is why it's usually a good idea to put the high power/gain stage right before your antenna. Even low-loss cables can cause you to drop significant power if they're too long or if you put them in the wrong place.

Gain is another frequency-dependent parameter that can be advertised in a misleading way. This is especially true for wideband amplifiers. No matter what the datasheet says, an amplifier can't be both perfectly wideband and perfectly high-gain. There will be a curve showing gain over frequency, and *that's* the chart you should look at, not the gain in huge print on the first datasheet page or in the ad. The gain is usually lowest near the edges of the advertised frequency range. For example, if a part claims that it has 30 dB of gain and works from 1 to 2 GHz, you probably won't see 30 dB of gain at 2 GHz. Very high-gain amplifiers will typically have a relatively small frequency range.

Amplifiers

If you want to amplify a signal up to any reasonably high power, you'll almost certainly need to use multiple amplifiers, or at least a single multistage amplifier. As such, don't be surprised if you can't find a part that will do, say, +30 dB of gain at +36 dBm out. Instead, maybe look for one amplifier that does +25 dB of gain at +26 dBm out, and a second amplifier that does +10 dB of gain at +36 dBm out. There's a trade-off between the maximum power out of an amplifier and the maximum gain it exhibits.

Don't count on being able to achieve the maximum power out advertised for an amplifier, however, especially if it's a lot of power. Give yourself at least 1 or 2 dB of headroom, even if you're going to be running the amplifier into hard saturation. As always, the best thing to do is to get an evaluation board of the amplifier you're interested in and drive it the way you'll be driving it in your product to see what performance you get. These development boards are often several hundred dollars, but you can usually get them for free from a sales engineer. You can also check eBay or AliExpress for unofficial development boards for some parts, although these usually don't come

with documentation. Manufacturers will also usually give you the Gerbers for the development board for free so you can fabricate and assemble it yourself. The only downside with that approach is that you don't get a piece of paper with measured, verified performance data of your development board from the manufacturer.

In an ideal amplifier, gain is linear. If you make the input signal a little bit bigger, the output will be proportionally bigger. But real amplifiers behave that way only up to a point, after which the amplifier starts to go into compression. The input power where you no longer get a linear gain, but instead get a gain 1 dB less than you would expect an ideal part to have, is called the *1 dB compression point*, or P_{1dB}. This is an important number to look at, especially if you're going to be running an amplifier in compression or at its maximum output power. Figure 4-1 illustrates the difference between an amplifier's ideal response and its actual response and shows the P_{1dB} point.

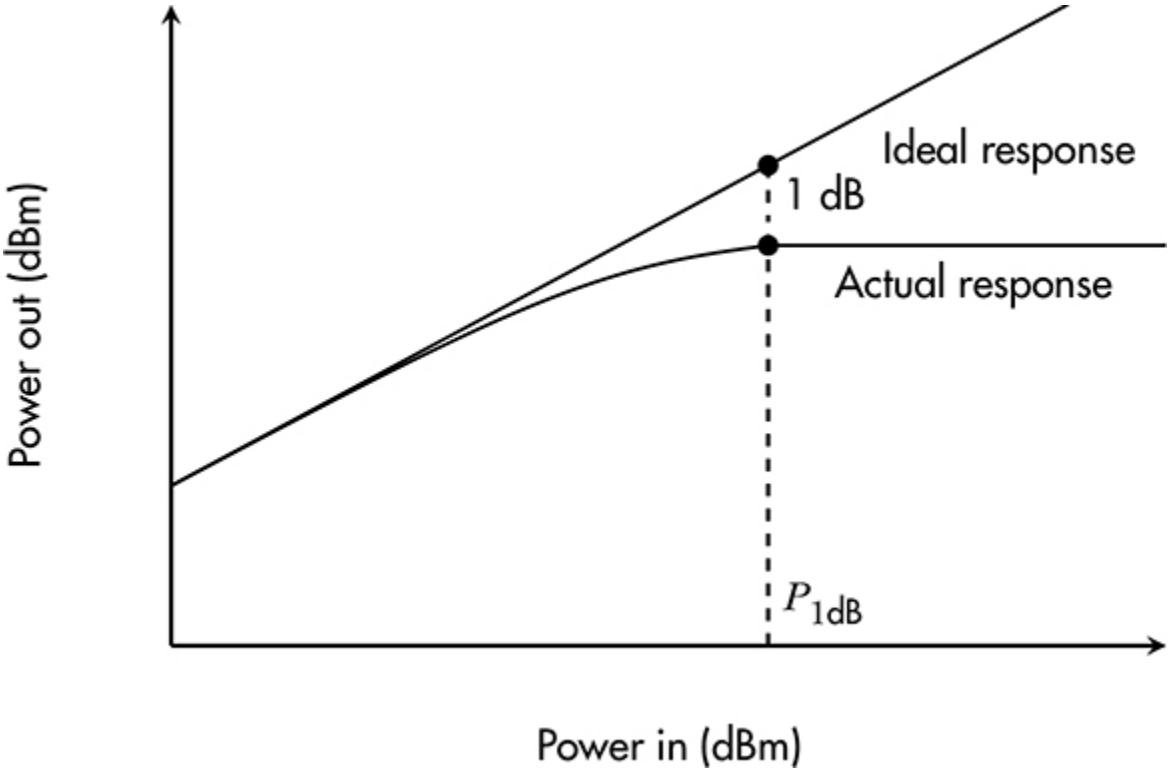


Figure 4-1: P_{1dB} is the point where an amplifier's gain is 1 dB less than it would ideally be.

Amplifiers also have a stability factor, or *k-factor*, that tells you if the amplifier will begin oscillating under poor impedance-matching conditions. You can find amplifiers that are both conditionally and unconditionally stable. Prefer amplifiers that are unconditionally stable. Unconditional stability means that no matter what impedance you put on the input or the output of the amplifier, it won't oscillate. That doesn't mean it will *never* oscillate (there are other conditions you can subject it to that can cause oscillation), but using an unconditionally stable amplifier can save you a lot of troubleshooting.

Why would anyone ever want to use an amplifier that *isn't* unconditionally stable? As in everything, there are trade-offs in making an amplifier unconditionally stable. Specifically, the noise figure or P1dB may be worse in an unconditionally stable design versus a conditionally stable design. If these are trade-offs you'd like to play with, then you can choose a conditionally stable amplifier.

NOTE

If you want to read more about amplifier stability, there's a classic paper called "Avoiding RF Oscillation" by Les Besser (Applied Microwave and Wireless, *spring 1995) with lots of good information. It's linked on this book's website*, https://designingelectronics.com.

Transistors

Discrete transistors find uses in devices like motor controllers, load switches, and power supplies. To use them successfully, there are a few factors you need to consider.

MOSFETs

MOSFETs (metal–oxide–semiconductor field-effect transistors) should always be derated. Follow the NASA recommendations shown in Table 4-2.

Table 4-2: MOSFET Derating Guidelines

Stress parameter	Derating
Power	60%
Current	75%
Voltage	75%
Junction temperature	80%
Gate-to-source voltage	60%
Source-to-drain voltage	75%

MOSFETs don't instantly turn on and off. They have a gate capacitance that can be quite large in some cases. This capacitance adds a time delay before the MOSFET turns on. When picking a MOSFET, check that the gate capacitance isn't so large that you won't be able to meet timing requirements. The equation for the time it takes for a MOSFET gate to charge to the gate voltage and turn on is:

$$t = R_G \times C_{iss} \times \ln \frac{V_{GS} - V_{TH}}{V_{GS} - V_{gp}}$$

In this equation, R_G is the gate resistance, C_{iss} is the gate capacitance of the MOSFET as seen by the gate driving circuit at V_{DS}, V_{GS} is the final gate voltage, V_{TH} is the gate turn-on threshold voltage, and V_{gp} is the gate plateau voltage. You can obtain all of these values except V_{gp} from tables in the datasheet. To get V_{gp}, find the gate charge curve and determine the voltage where the curve plateaus. The gate charge curve will have V_{GS} on the y-axis and Q_g on the x-axis. Figure 4-2 shows a typical curve.

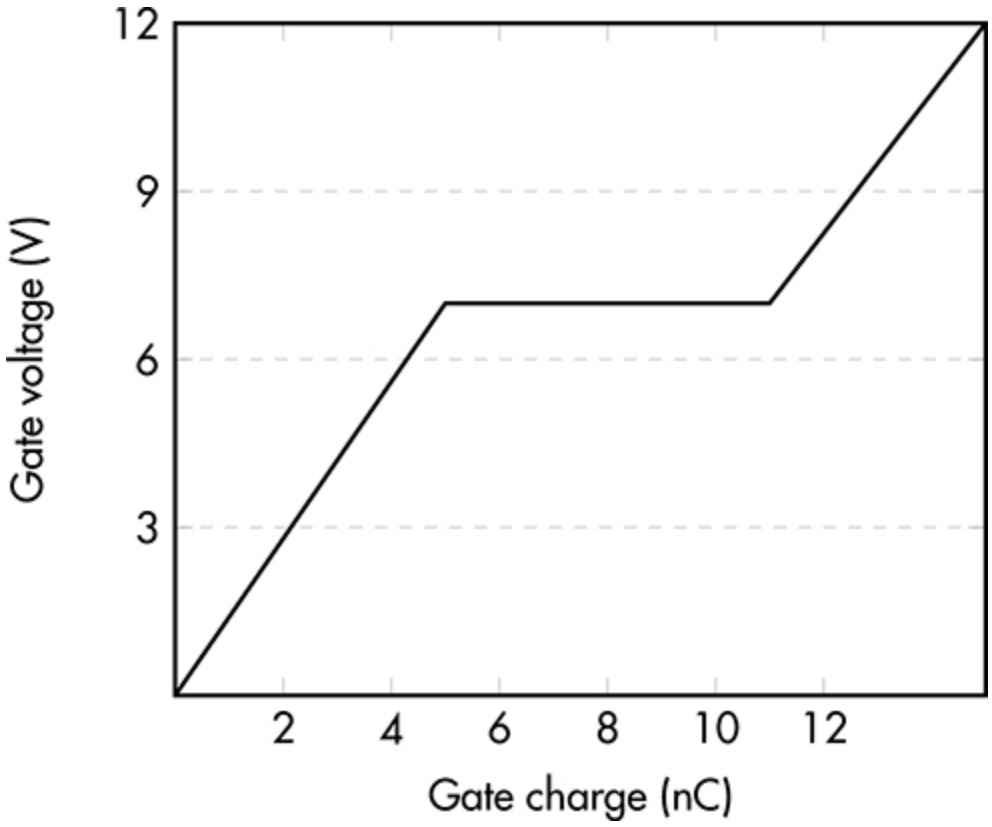


Figure 4-2: Example gate charge curve

In the example shown in the figure, the plateau occurs in the middle of the curve, at about 7 V. That's the value of V_{gp}.

NOTE

The full derivation of the gate charge time equation can be found in Vishay application note AN608A, available at https://www.vishay.com/docs/73217/an608a.pdf.

Drivers

Determine if you need a driver IC or circuit to control any switching semiconductors. For example, a microcontroller or processor general-purpose input/output (GPIO) pin won't be able to provide enough current or voltage to fully switch a large power transistor. GaN FETs require a specific power-up sequence or else they'll break, so you need a special driving circuit for those devices. When choosing the driving

method, pay attention to the switching times so that you can meet timing requirements. Having a fast MOSFET doesn't do you any good if you use a slow driver.

Drain-to-Source Resistance

All diodes and FETs dissipate power because there's a nonzero drain-to-source resistance, or R_{ds}. If you're designing low-power or battery-operated electronics, a high R_{ds} will dissipate power unnecessarily. This can cause your measurement of consumed power to be higher than the value you calculated. In high-power devices that are switching large currents, a high R_{ds} can cause your MOSFETs to overheat and fail prematurely or require significant thermal management.

Avoid these problems in the first place by checking the drain-to-source resistance during component selection. You can calculate how much power (in watts) your MOSFET will dissipate as a result of R_{ds} by simply squaring I_{ds} and multiplying it by R_{ds}. It's typically pretty easy to find a MOSFET that has an R_{ds} of under 1 Ω, so that's a good place to start.

It's important to note that R_{ds} increases as temperature increases, so if you're running your circuit in a hot environment or if you have insufficient heat sinking, your MOSFET will get hot, which will cause R_{ds} to increase. This will dissipate more energy as heat, which will make R_{ds} increase even more. You'll get stuck in thermal runaway until your MOSFET explodes. To avoid this, make sure that you calculate the expected thermal dissipation due to R_{ds} across your entire temperature specification.

Diodes

Choose LEDs to be sufficiently bright. LEDs should have a maximum forward current of less than the maximum current source and sink rating on the LED driver or GPIO pin you're using to drive them. If

they don't, they should be buffered. You can't run an ultrabright 10 W LED off a microcontroller pin.

When using a diode in series, make sure you compensate for the voltage drop across the diode. If you need a really small forward voltage drop across a diode (usually the case when you use a diode in series), use a Schottky diode. The trade-off is that they have a higher reverse bias leakage current, and this gets worse as the Schottky diode's temperature increases.

Make sure the diode you pick can handle the current you need to put through it. You also need to specify the reverse voltage (the point at which the diode will begin to break down and start conducting in the "wrong" direction). Once you start approaching the rated reverse voltage, the diode will start to get leaky, and small amounts of current will start to flow. Use the I-V curve in the datasheet of the part you pick to figure out exactly how much current that is. If you don't give yourself enough headroom between the rated values and what you're actually subjecting the part to, you can end up with an inefficient design, since you'll be losing energy through diode leakage.

It's possible to use a Zener diode as a voltage regulator, but it's not recommended. A Zener diode regulator will have poor efficiency and poor power handling, and it won't be very good at maintaining a regulated voltage. That said, they're very cheap, and if you have a low-power application that can handle those drawbacks, it may be an okay choice. A related application of Zeners that can work well is as a voltage reference. You can also use a Zener as part of a clamping circuit to ensure that the voltage at a particular point never exceeds the reverse breakdown voltage of the Zener diode.

Be aware that low-voltage Zeners (below about 6 V) have a soft knee —that is, they start conducting before the breakdown point, making them look "leaky." Zener diodes are also temperature sensitive. In fact, they have a negative temperature coefficient when operated below about 5 V and a positive temperature coefficient when operated above 5 V. The datasheet will usually contain a graph that shows you the temperature coefficient versus the voltage.

Batteries

There are dozens of battery chemistries to choose from, and each one has its own advantages and limitations. Which chemistry you choose will depend on what fulfills your requirements, but 90 percent of designs can get by with one of the chemistries discussed in this section. If you're in the 10 percent that's unable to use any of these chemistries, you either need a more exotic chemistry (which is outside the scope of this book) or, more likely, need to try to change your requirements or specifications.

Remember that you need to properly dispose of all battery types. If you just throw batteries away, they can poison the environment at best and explode or start a fire at worst.

If you need to measure detailed information about your battery performance, you can use a chip called a *gas gauge*. Gas gauges are all slightly different, but generally they have a Coulomb counter to measure charge accumulation into a battery, as well as current and voltage sensing. They can also sometimes provide digital outputs that report whether the battery voltage is sufficiently high, the charging state, and other useful pieces of information for managing battery charging or alerting your user.

Note that in battery parlance, a *primary battery* isn't rechargeable, while a *secondary battery* is rechargeable. A battery's capacity is measured in amp-hours (Ah) or milliamp hours (mAh). A battery with a rating of 1 Ah can discharge 1 amp for 1 hour, 0.5 amps for 2 hours, 2 amps for 0.5 hours, and so on.

Batteries also have a *C rating*, a dimensionless number used to describe the battery's charge and discharge rate. When a battery's C rating is listed, it's usually referring to the maximum discharge rate. For example, a 1 Ah battery with a 1 C rating means that it can discharge no more than 1 amp in 1 hour before damage starts to occur. The same 1 Ah battery with a 10 C rating means that it can discharge up to 10 amps for 0.1 hours (which is 6 minutes). However, just because you *can* discharge a battery at a huge C rate doesn't mean you should. Pulling a lot of current from a battery is going to make your battery get hot and

will probably shorten its lifespan if you do it over and over. It's better to make sure you have a little room between the C rate you're using and the maximum C rating of your battery. Also, never charge batteries at a C rating above what the datasheet recommends. For most batteries, this is 1 C. Doing so could damage the battery.

Lithium

When people talk about lithium batteries, they're usually referring to secondary (rechargeable) lithium-ion batteries. Within the group of lithiumion batteries, the two most common chemistries you'll encounter in consumer electronics are lithium polymer (LiPo) and lithium iron phosphate (LiFePO4).

In lithium polymer batteries, the electrolyte is a polymer instead of a liquid. These batteries have a high energy density and a nominal cell voltage of 3.7 V. The two most ubiquitous form factors of LiPo batteries are flat cells (also called prismatic cells) and cylindrical cells. Flat cells are exactly what they sound like: thin, rectangular batteries. They come in all sizes, and cells of this kind are typically stacked on top of one another for multicell packs. Meanwhile, cylindrical cells come in standard sizes with the names referring to the physical dimensions of the battery. For example, the 18650 is one of the most common and popular cylindrical cells and has a diameter of 18 mm and a length of 65 mm (just ignore the ending zero). There are other sizes of cylindrical cells available, but 18650 is by far the most common and popular. In addition to these two common form factors, LiPo batteries are also available as coin cells. There are many different sizes of coin cells, and you can buy both primary and secondary lithium coin cells.

LiPo batteries with multiple cells must be balanced. That means the voltage on each cell needs to be the same so they're all discharged at the same rate. Many multicell LiPo batteries come with balancing circuitry already included in the pack. Depending on how the battery is constructed, however, you'll probably still need to use a balancing external charger.

The failure mode you'll see if the batteries aren't balanced is that the pack will begin to get puffy as gas escapes and builds up inside. If a LiPo

is puffy, stop using it. Puffy LiPos can heat up, self-ignite, and then self-oxidize. Because the battery creates its own oxidizer when burning, it's impossible to put out the fire with water. You just have to sit back and wait for the fire to burn itself out. Some battery packs have temperature probes built into them so the charger can detect any failure modes before they happen and stop charging. It's a good idea to use this feature if you can.

There are many cheap ICs designed explicitly for charging LiPo batteries, from one to many cells. This way, the battery can be charged while it's installed inside the device, rather than the user having to remove the battery and charge it externally. These charging ICs do more than just apply a voltage and balance the cells. They also change the voltage and current curves going into the battery to optimize its lifetime and charge it as quickly and safely as possible.

Lithium iron phosphate is a chemistry that has the advantages of LiPo while being much safer. When LiPo cells are shorted, they heat up and can cause a fire. LiFePO4 batteries won't catch on fire when punctured or crushed. The trade-off is that they have slightly less energy density than LiPo batteries and can cost slightly more.

Another new lithium chemistry that's now commercially available is called lithium thionyl chloride. These batteries are primary (not rechargeable), but they can have an extremely high energy density. A battery of this chemistry the size of a D cell alkaline battery can hold up to 20,000 mAh at 3.6 V.

Regardless of the type, lithium batteries should be stored at around 40 percent charge to maximize their lifetime. Most consumer products that use lithium batteries will be packaged and shipped at about 40 percent to 70 percent charge. However, it's important not to allow the cells to discharge below 2.0 V per cell or the battery may be permanently damaged. Note that if you're shipping batteries, there may be requirements for the level of charge that is allowed for shipping, depending on where you are shipping to and from and if the batteries are on their own or are inside a product.

Lead Acid

Lead acid batteries are almost never used in consumer electronics, but they are sometimes used in robotics, automotive engineering, or other contexts where you need a very large energy density and a large current surge but don't care much about battery size or weight. Lead acid batteries are very heavy (since they contain mostly lead) and have a cell voltage of 2 V. You usually see 6 V, 12 V, and 24 V lead acid battery packs.

Lead acid batteries are relatively safe. However, they do contain acid, and they also release hydrogen gas when they're charging, which can explode under the right circumstances. They're easy to recycle, provided you don't allow the lead to leach into the environment. To prolong their lifetime, lead acid batteries should be stored at 100 percent charge.

Alkaline

Alkaline batteries are primary batteries, meaning they're not rechargeable. This is the chemistry that AA, AAA, C, and D batteries use. Alkaline batteries are the cheapest way to implement battery power because they don't require recharging circuitry. However, they do require extra mechanical engineering, since your user needs a way to replace them when they die, whereas a rechargeable battery can be charged just by exposing a small connector somewhere on the device. Keep in mind that removable battery covers can cause problems with dust or water ingress into the device.

Correctly stored, a primary alkaline battery can have a shelf life of 10 years. Because alkaline batteries are primary cells, you don't need to discharge them at all before you store them. There will be some self-discharge over time, but it will be fairly minimal. Like lithium batteries, alkaline batteries are also available as coin cells (although not all coin cells are alkaline; some use different chemistries).

Conclusion

Active components are what give your product the features it needs. They're also probably the most expensive parts in your design. They

consume the most power and get the hottest, too. Choosing and using your active components is a key part of the design process. In this chapter, we looked at all kinds of active components and what to consider when choosing them.

PART II
DESIGNING

In Chapters 5 through 9, you'll learn how to engineer your product to work as you expect. We'll talk about clear communication in your schematic, PCB layout tips to ensure your design works, and things to consider so that your design is easy to manufacture at scale. You'll also learn how to avoid common mistakes that cause designs to fail regulatory testing, as well as methods to reduce the cost of your design.

5
SCHEMATIC DESIGN

Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Readability counts.
—Tim Peters, *The Zen of Python*

Your schematic is your primary design document, and its clarity is crucial to ensuring your device's success. It must be correct, and it must also be easy for humans to read and understand. In this chapter, we'll delve into the best practices for schematic design and explore how to avoid common mistakes. We'll also discuss ways to optimize your schematics for debugging and performance.

Conventions

There's no one right way to draw a schematic, but following certain rules and conventions will help keep your schematics understandable and reduce the likelihood of design mistakes. You don't *have* to observe all of these conventions—maybe you're used to using different letters for component designators, or your EDA software doesn't behave in the

way recommended here—but following the guidelines discussed in this section will make schematic capture and debugging easier.

Labeling and Organization

You need to label every component in your schematic with a unique designator. It should start with a letter indicating the class of component, such as C for capacitor or R for resistor, followed by a unique number. The American National Standards Institute (ANSI) and the American Society of Mechanical Engineers (ASME) have a joint standard for these letter prefixes. Confusingly, IPC also has a standard that is different from the ANSI/ASME standard. On top of all of that, there are many companies that don't follow either of these standards. The good news is that both standards and the versions that different companies use are all pretty close. Table 5-1 contains a list of letter prefixes that isn't standard-compliant but matches what's most commonly seen in the industry. You don't have to follow this convention exactly, but make sure you do follow *some* convention consistently.

Table 5-1: Preferred Component Prefixes

Prefix	Component
A or AR	Amplifier
B or MG	Motor or motor generator
BT	Battery, battery cell, or solar cell
C	Capacitor
CB	Circuit breaker or network protector
D	Diode, Zener, Schottky, transient suppressor, LED
DS	Display, lamp, LED, visual indicator
E	Antenna or miscellaneous electrical part
F	Fuse

Prefix	Component
FD	Fiducial
FL	Filter
H	Hardware, mounting screw, and so on
J	Connector or jack
K	Relay or contactor
L	Inductor, coil, or choke
M	Meter
MH	Mounting hole
MK	Microphone
P	Connector, mechanical feature, fiducial, screw hole
PS	Power supply
Q	Transistor, FET, SCR, TRIAC
R	Resistor
S	Switch
T	Transformer
TB	Terminal board or terminal strip
TP	Test point
U	Integrated circuit
VR	Voltage regulator
W	Wire, cable, or cable assembly
X	Holder for a fuse, lamp, or battery
Y	Crystal or crystal oscillator

Most EDA software will automatically handle component designation. Some CAD packages will assign a unique designator every time you put down a new component. Others require you to explicitly run a designator assignment.

Just as every component in the schematic has exactly one unique designator, every designator must be mapped to exactly one component in the layout. There should be no components in the layout that aren't in the schematic. Depending on the EDA software you use, you can usually check for this by running an electrical rule check (ERC). Some tools will make a huge fuss if the schematic and layout ever get out of sync. Other tools work on a different model and purposely let the schematic get ahead of the layout. Then you have an update step in the workflow, where you tell the software to add the new parts and nets in the schematic to the layout.

If you have a large component with many pins, it may be impractical to draw a single schematic symbol for it. Instead, break it up into multiple blocks, or subsymbols. To indicate this, append the component's designator with a different letter for each block. For instance, a chip that has two schematic subsymbols might be called U3A and U3B. Try using the subsymbols to group similar pins together regardless of the pins' physical locations on the real chip. You might have one subsymbol for power and ground pins, one for analog pins, one for digital pins, and so on. This will help make the circuit easier to follow. For example, Figure 5-1 shows part of a schematic with one subcomponent of an FPGA, labeled U3F. The subcomponent groups together just the power pins on the device, even though those pins are physically spread all across the chip.

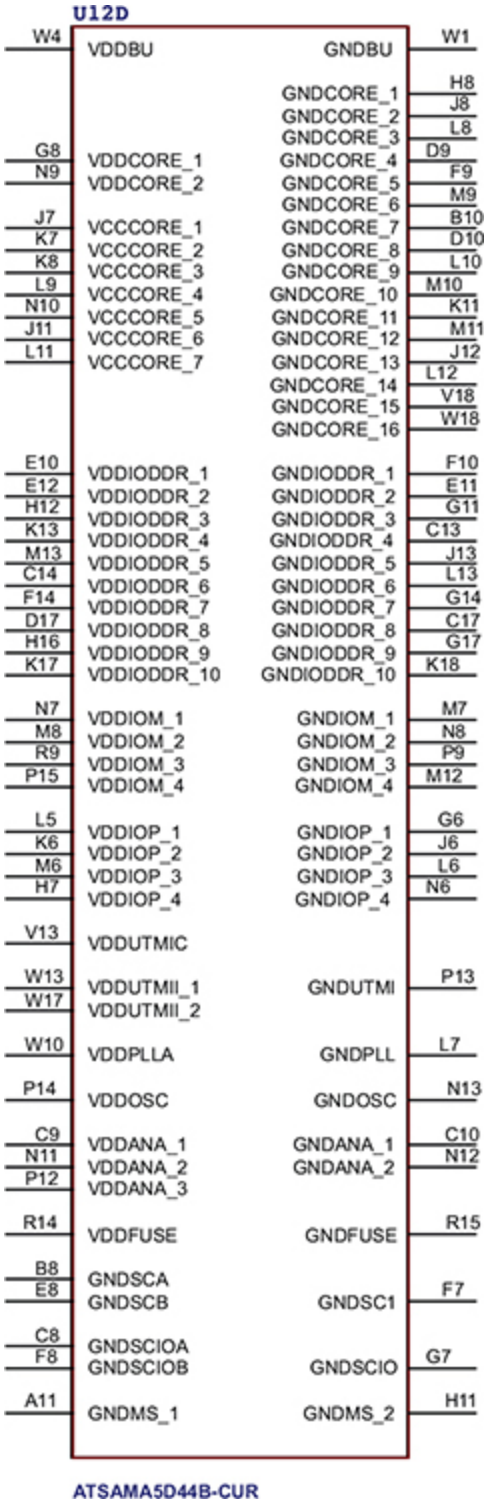


Figure 5-1: The FPGA in this schematic has been broken up into multiple blocks, or subcomponents. This subcomponent shows the power pins on the device.

Figure 5-2 shows another FPGA subcomponent (U3D) from the same schematic. This one features a single bank of I/O pins. It's much

easier to find the I/O pin you're looking for in a subcomponent block like this than to hunt around the edges of an enormous component symbol with power, I/O, and other pins scattered throughout.

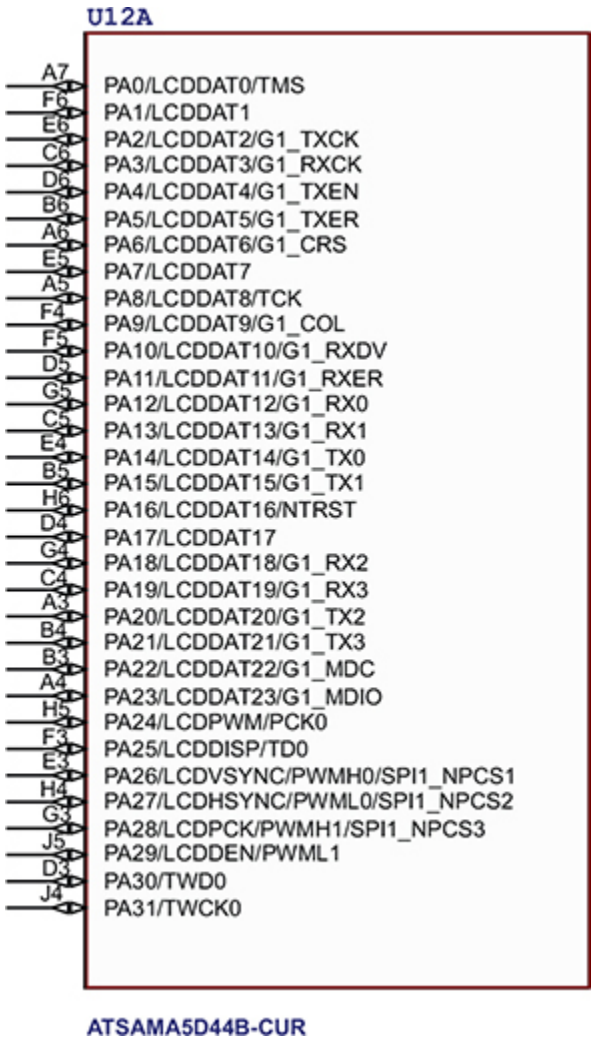


Figure 5-2: This other block from the same schematic as Figure 5-1 shows an FPGA subcomponent containing a bank of I/O pins.

Show the values of each component next to its schematic symbol. Use the suffix R to mean ohms when specifying resistance (the letter O can get confusing because it looks like a zero). If the resistance is greater than 1,000 Ω, you can just use the letter K without the letter R. For example, you'd write a 100 Ω resistor as 100R and a 1,100 Ω resistor as 1.1K.

If a component is unusual or must come from a particular manufacturer, make the manufacturer part number visible next to the schematic symbol as well. This will mostly be useful when someone else is reading the schematic. They'll be alerted to the fact that the part is unusual and will be able to copy and paste the part number into their browser to get the datasheet for it.

With the exception of two-pin passive parts (like inductors, capacitors, and resistors), don't hide any pins or pin numbers in the schematic symbol. Your reviewers will assume the pinout is correct or forget to go into the library for the part and check. When the pins and numbers are visible, it's simple to pull up the datasheet and check against the pinout and application circuit.

Footprint mistakes are all too common in the first iteration of a PCB. It's easy to mess up the mapping from signal to pad because there are several conversion steps in between that depend on different conventions. There are the pin numbers on the schematic symbol itself, the pin numbers on the pads in the footprint, and finally the mapping between the two. To avoid confusion, make the pin numbers on the schematic symbol the same as the pad numbers in the footprint so the mapping is always 1 to 1, 2 to 2, and so on.

The one case where this pin mapping won't work is with BGA components, where pin names use both letters and numbers. For example, if the first row of pins on the footprint is labeled A1, A2, A3, . . . then you can number those 1, 2, 3, . . . on the schematic symbol. Once you get to the second row, where footprint pads are named B1, B2, B3, . . . you'll have to name them 4, 5, 6, ... This can get confusing and it's easy to make a mistake, so take extra care if you have to do this. If your CAD tool allows you to use letters in your pin numbering, I recommend keeping the schematic-to-footprint mapping the same: A1 to A1, B1 to B1, and so on.

Problems arise when you try to reuse a schematic symbol for a new footprint. This most commonly happens with connectors. You already have a schematic symbol drawn for a 20-pin connector, but you have multiple physical connectors that have 20 pins. *Usually*, connectors and ICs use the same pin numbering scheme, but not always! We'll discuss

avoiding mistakes with connectors in more detail later in the chapter, but for now, just know that keeping a one-to-one mapping between all pin numbers and pad numbers could save you a board spin or painful rework. Always manually check the mapping on your connectors. The best way to avoid the schematic symbol reuse problem is not to reuse schematic symbols. It doesn't take that long to draw a new one, and you'll know it's correct.

If you're using a polarized capacitor, your schematic should indicate which side is the anode and which is the cathode. The most common way of doing this in the US is to make the line on the cathode side curved ("C for cathode" is a good way to remember this), as shown on the left of Figure 5-3. In Europe, you'll often see the symbol on the right. You can remember this with another mnemonic: The *colored-in* side is the *cathode*.

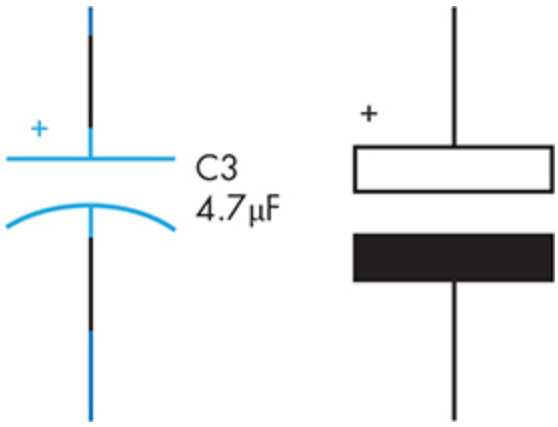


Figure 5-3: American-style (left) and European-style (right) polarized capacitor symbols

The curved line or colored-in side works for visually denoting polarization, but if you get your footprint pin numbering mixed up, the part can still be populated wrong. Even if you're assembling by hand and pay attention to polarity, any future assembly jobs that are handled with a pick-and-place machine might get it wrong if the polarity is swapped in CAD. Some designers will set the pin names to A and K (anode and cathode) in both the footprint and the schematic symbol to guarantee that the polarity is correct. This also works with diodes, which similarly have only two pins and a polarity. The advantage of this approach is that you can reuse schematic symbols with less risk, since

you're matching A to A and K to K. It works even if different components use different conventions for where pin 1 is.

Supplemental Information

A schematic can and should contain more than just the circuit diagram. Just as software engineers are encouraged to add comments to their code explaining how the program works and why it's been implemented a certain way, your schematics should likewise be annotated so it's obvious what's going on and why you made your design decisions. This will help other people read and review your schematic, not to mention your future self when you forget why you put that resistor there or how you calculated that value.

Include details like critical layout, shielding, and grounding requirements. You'll usually do schematic capture before layout, so anything that will be helpful during layout should go in the schematic. Text boxes and comments work well for this, but the schematic itself can also describe graphically how to lay the circuit out. For example, when placing the capacitors for a particular IC near the chip itself in the schematic, try placing smaller-value capacitors next to the IC symbol and larger capacitor values farther away. This visually represents that capacitors should be placed the same way during layout. Don't try to make the schematic look like the desired layout at the expense of schematic clarity, however.

Another useful supplement is to call out test points on the schematic and include information that may be helpful in debugging or troubleshooting faulty units. For example, you can note what the expected waveform, voltage, current, or frequency should be at various points, especially test points, as in the old-school schematic shown in Figure 5-4. This is really helpful during assembly, testing, and debugging.

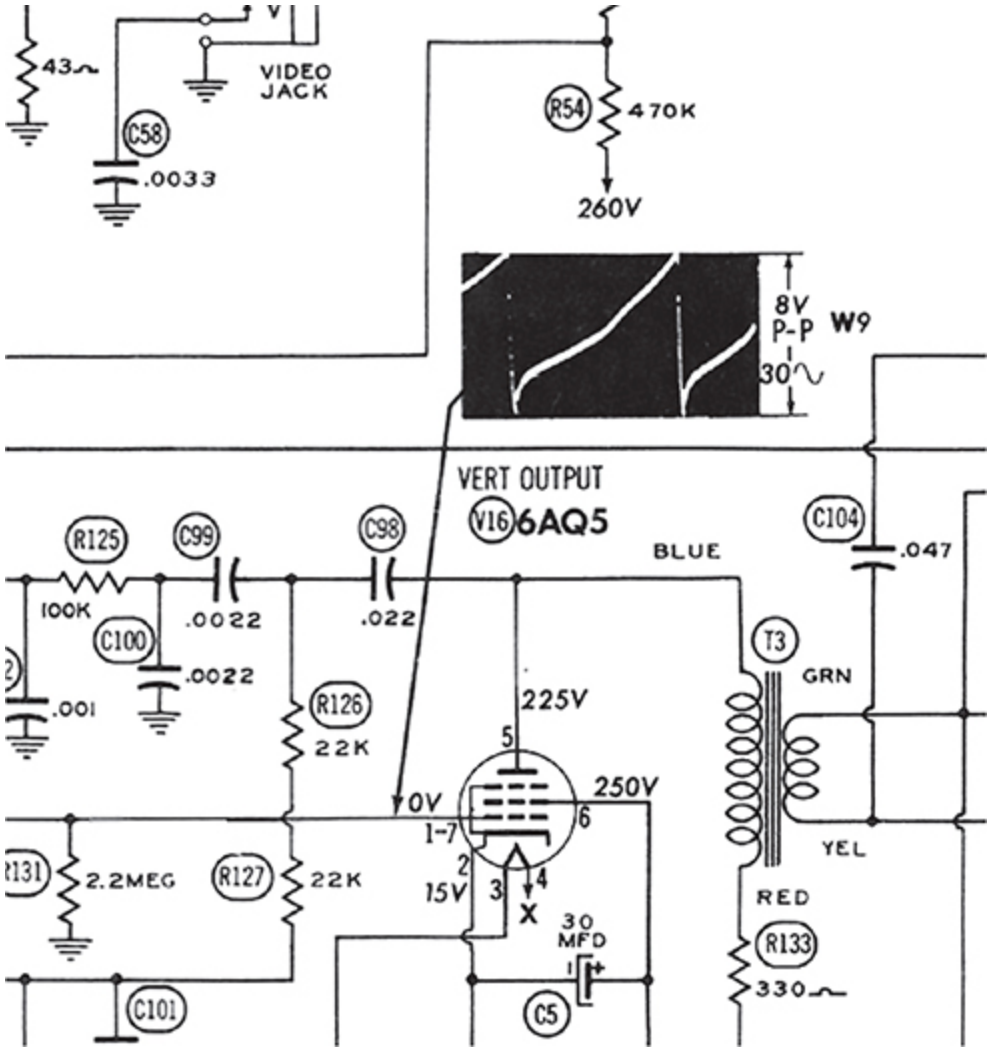


Figure 5-4: Part of an old television schematic from the 1950s, including an image of the expected waveform at a certain point in the circuit

Keep in mind that while supplemental text is a great way to annotate and clarify a schematic, this text shouldn't overlap with any wires or symbols. Part-specific detail should be placed as a visible component parameter, not as free text. That is, instead of dropping in a text box next to the component, go into the component properties, type the text into the comment field, and make the comment visible next to the symbol. Not all EDA software supports this, but if the option is available, it's easier to generate documentation if the software knows which note goes with which component. It's also useful if you need to move the part later, since unlike a text box, the comment will move with it.

A high-level block diagram of your circuit is a valuable supplement to the schematic itself. If you use hierarchical blocks in your schematic, you can combine the two so that your high-level schematic acts like a block diagram showing where signals go. Then opening the schematic represented by each hierarchical block shows you the details of how that block is implemented.

Formatting

When drawing a schematic, you'll often see schematic symbols with inputs on the left side, outputs on the right side, power on the top, and ground on the bottom. However, sometimes it's more beneficial to group pins by function. You can group all address lines together, all control lines together, or all power pins together to create a more legible and clearer schematic. Just drawing the schematic symbol exactly the same as the physical pinout can result in a rat's nest of wires on your schematic.

Power and ground symbols don't cost you anything in your schematic, so use them liberally. Don't draw large loops of wire to connect power and ground; just add another symbol. Ground symbols should always point down and power arrows should always point up so they won't be mistaken for another component.

Always place dots on all net junctions if they're meant to be connected. If two wires cross each other in the schematic and there isn't a dot at their intersection, those wires shouldn't be electrically connected. Better yet, try to avoid crossing unconnected wires whenever possible and practical so there's no ambiguity as to whether a junction dot was accidentally omitted. It's better to use "T" junctions instead of "+" junctions (sometimes called fourway junctions) so it's easier to find where the EDA software mistakenly forgot a junction dot and thus didn't make an electrical connection.

Make sure to use a consistent grid when drawing your schematic symbols and when drawing the schematics themselves. If you draw a symbol on a grid that uses different units or a different spacing than the overall schematic, you can end up with slightly offset or off-grid parts in

your schematic. At best, it looks bad. At worst, it can make a wire next to a pin look connected when it's actually not.

Don't feel pressure to keep the entire schematic on a single page. You or someone else will probably print out the schematic at some point, so make sure a single page isn't so crowded that it's difficult to read or really tiny when printed out. Put subcircuits on their own schematic pages. You can organize multipage schematics in two main ways: hierarchical design blocks or flat sheets with inter-sheet references. Flat sheets are simpler and work well for more basic designs. Hierarchical blocks are better suited for large, complex designs, especially those that feature many instances of the same circuit. Personally, I prefer hierarchical blocks, and if you're using Altium, they're supported quite well.

Try to keep all schematic sheets the same size so it's easy to print them out on a normal printer. In the US, most people are familiar with 8.5×11 inches, but that isn't always big enough to fit a schematic. The next size up you should use (in the US) is legal paper, which is 8.5×14 inches, and after that is tabloid, which is 11×17 inches. I like to use tabloid paper when printing schematics out. Whatever size you use, try to be consistent and pick a size that allows you to adequately display the schematic without it being too dense or too small to read. Remember, readability is paramount.

For neatness and completeness, put the name of the circuit, its version, and the project on each page. Schematic sheets typically have a small box in the lower-right corner for this purpose. When you print schematic pages out and have them lying all over your desk, you'll be thankful that you can tell which page goes where and whether it's an old version.

Make sure to name your nets; this will make layout a lot easier. Use a standard naming convention for special nets. Here are some common examples: Differential pairs have _N or _P at the end of the net name, _L means the signal is active low, VDD_3V3 means the net is a 3.3 V supply rail, CLK_ prepended to the beginning of the net name means it's a clock, and LX_ prepended to the beginning means it's a switching node in a DC/DC converter. Having these predictable net names makes

it easier to apply design rules or signal integrity constraints in your CAD software. Sticking with a consistent convention can also help avoid the mistake of accidentally isolating two parts of the schematic that are supposed to be connected because you used two different names. For example, a net called "5.0V" and a net called "+5V" won't be connected.

Review Checklist

The best way to catch mistakes before you have other people review your design is with your EDA software's ERC. This will automatically check for single-ended nets, unused I/O, unterminated signals, and other common errors. The goal should be to have zero ERC errors and warnings. If you're getting warnings about unused pins, for example, add pin terminations or ERC ignore markers to them. If you *do* end up with some ERC warnings or errors, you need to be able to explain every single one, why it's okay, and why you didn't fix it.

Beyond relying on the software, the following tips will help you avoid mistakes during the schematic design process:

Check the supply voltage. Make sure every part will be running on its correct operating voltage. This is a really simple and seemingly obvious thing to check, but when you're working with dozens of chips that run at different voltages, it's surprisingly easy to mess up.

Check your logic levels. It's easy to connect a 3.3 V GPIO to a 5 V part without realizing it. If the 5 V part doesn't have a logic threshold of at least 3.3 V, you won't be able to control it with a 3.3 V GPIO. You need to either use a level-shifter IC or make one using a buffer. Whichever solution you use, make sure the level shifting will work bidirectionally if necessary.

Get the latest datasheet. There are often multiple versions of a part's datasheet floating around online, so it's important to make sure you have the most up-to-date copy. Always download datasheets from the product page on the manufacturer's website instead of

simply googling a part number and downloading the first version of the datasheet you find.

Verify GPIO special function assignments. Some microcontrollers only support certain features on specific GPIO pins. ADC inputs, timer inputs and outputs, UARTS, and interrupts are usually limited to a fixed range of GPIO pins. If you're using one of these features on a micro-controller, make sure the pin in question will allow it.

Don't forget pull-up resistors. Any open-collector or open-drain output should have a pull-up or pull-down resistor to keep it in a known state at all times. Floating outputs can cause problems with other devices or interfaces. I2C buses need a pull-up resistor on both the SDA and SCL lines.

Include necessary protection circuitry. For example, if you're driving anything that's strongly piezoelectric, include protection circuitry for the driver so that any large mechanical impulses don't cause a voltage spike and destroy the driver. Likewise, if you're designing an application that uses an H-bridge or drives another large inductive load, make sure you use a flyback diode so current can flow only in one direction. Otherwise, the inductor will produce a large voltage to oppose the changing current running through it. This means that when you switch current through a motor (which looks like an inductor), a large voltage will appear across the motor, which can destroy your driving circuit.

Make your circuit hard for people to break. Polarized capacitors aren't safe to be reverse-biased, so design your circuit such that you know this won't ever happen. Reverse polarity protection for the entire design is a good idea, whether that's through electronic protection that can survive a cable being plugged in backward or through mechanical means that make it impossible to plug a cable in backward. If your device can act as a USB host, consider adding overcurrent protection to limit the current draw on the 5 V rail. Misbehaving USB devices may try to draw more current than you specified or fail short.

Calculate a power budget. Every power supply should be able to provide enough current and power for the components it's powering. Leave yourself some wiggle room. Have you compensated for any voltage drops across diodes on your power supply or from LDOs? Think about efficiency, too: You don't want to be burning a huge amount of power as heat in linear regulators. Large voltage conversions are better accomplished with a switching supply.

Calculate a link budget. Leave yourself some wiggle room here as well. Every wireless link should work without SNR problems because you calculated the gain and loss of each element in the chain. Estimate the channel loss and give yourself plenty of room. Don't forget that decibels are a logarithmic scale. It's easy to shrug off 1 or 2 dB; however, 1 or 2 dB when you're talking about multiple watts of power is significant. Converting to watts can help give you a more intuitive idea about whether those couple of decibels matter.

Consider all battery conditions. If your design is battery-powered, make sure everything will keep working even when the battery is discharged to its minimum operating point. Your product specifications will indicate where this point is. LDOs are particularly sensitive to battery levels. If you have an LDO with a 5 V output, it may need at least 5.1 V of input (or more). If you're running that LDO directly off your battery, that means your circuit will power off when the battery drops below 5 V. You may need to add a boost converter to keep that voltage high enough even after the battery has discharged below the LDO's power threshold.

Anticipate race conditions. These can happen in hardware just like in software. Look for any cases where the order of arrival of signals matters. Using pull-up and pull-down resistors can help prevent these cases. A technique called *back pressure* borrowed from FPGA development can be useful, too. As data moves along a pipeline, each stage waits for a "data ready" signal from the previous stage before trying to receive and process the data. This is often applicable to power supplies, some of which have "power good" pins that will tell you when the output is stable and ready for use. Using

back pressure can help prevent cases where one stage or module in a design is activated before the previous stage has finished.

Check your connectors. It's all too easy to look at a datasheet drawing, confuse the top-down view with the flipped-upside-down view, and reverse all the connections, so make sure all connectors and their mates are in the right direction, orientation, and gender. It's best to print out the layout at a 1:1 scale, label pin 1, lay the real connector on the piece of paper, and make sure it's facing the right direction and will mate correctly. Don't just think about it in your head; actually do it.

Don't forget unused pins. If you have any unused inputs or outputs, make sure it's okay to leave them floating and unterminated. The data-sheet will tell you if you need to connect unused pins to ground. If it's okay to connect an unused pin to ground, you should do so. This can actually reduce the inductance of neighboring pins, since the signal doesn't have to go as far to get to the return path. This may reduce signal-integrity issues.

Imitate development boards, but be careful. If you find a discrepancy between the design in a part's application note and the design of the part's development board, prefer the development board design—but only if you've tested the board and it performs as you want. Development boards are often designed by junior application engineers, and there's a decent chance that they won't perform optimally. Worse than that, there's an 80 to 90 percent chance that the application note design won't meet EMI and EMC requirements. Development boards are almost never FCC certified, so if you copy and paste the layout from the development board into your design, there's no guarantee it will pass. Where development boards are useful is when you're trying to verify component performance. You can usually request measurements of the part from the manufacturer to verify that you get the same results on your copy of the board. This will let you verify an application schematic you can use in your design, but you should expect to have to modify the layout to be able to use it in a real product.

Debounce. If you're using a potentiometer, keep in mind that the wiper may bounce open momentarily as you turn it. That means your circuit needs to be able to survive the potentiometer looking like an open circuit for at least a couple of microseconds. Push buttons and switches also bounce around before settling on their position. There are lots of examples of debouncing circuits that can address this problem. Don't forget that your software must be able to survive this short period of bouncing as well; if a program polls the status of the switch or potentiometer before it's finished bouncing around, it may read the wrong value. Look for debounce code that can solve the problem without any additional hardware.

Check that the parts on your BOM are in stock. Do this when you're starting a design, and again after you complete your layout. You can lose valuable time if you get far into the design process before realizing that one or more of the parts you were banking on using aren't available and that you need to redesign without them.

Terminate transmission lines. High-speed and RF lines need to be terminated at the proper impedance. The right way to do this will depend on your application, but it's important not to forget.

Don't mix up TX and RX. All UARTs should have TX and RX connected correctly. This sounds simple, but it's complicated by the fact that sometimes signals are labeled with *their own* name, and sometimes they're named by what they *connect* to. That is, sometimes the signal labeled TX goes to the pin labeled RX, like you would expect, but sometimes the signal labeled TX goes to the pin labeled TX. Read the datasheet carefully and make sure you're connecting a transmitter to a receiver, not a transmitter to a transmitter or a receiver to a receiver. Consider naming nets something more descriptive, like TX_DO for "TX data out" and RX_DI for "RX data in." You can also place a 0 Ω resistor in series with each trace so that if you make a mistake, you can remove them and solder jumper wires to fix it.

Check your reset polarity. If something won't program, it might be because it's being held in reset. Check that you haven't put a pull-

up or pull-down resistor on a reset line that pulls it the wrong way. If you decide to give yourself the option of both a pull-up and pull-down resistor, make sure one is marked as DNP and that the BOM reflects that. Similarly, make sure all enable pins are at the correct logic level and have a pull-up or pull-down resistor.

You may already be aware of all of these mistakes, but they're easy to forget about. Treat this section like a checklist that you can go through before you send your Gerbers or if you're reviewing someone else's design.

Debugging

It's important to plan ahead for debugging during the design phase. You're reading this book, so nothing will ever go wrong in any of your designs again, but *just in case*, make the troubleshooting process easy on yourself by incorporating debugging mechanisms directly into the design. For example, add test points to important signals, and put ground pads next to those test points so it's easy to clip on an oscilloscope probe without stretching the ground wire across the entire board. This placement isn't just a matter of practicality; it also ensures signal integrity and accurate measurements. If you don't want to populate actual test points, just use the exposed edge of a nearby passive on that net or drop in a 0 Ω resistor in series with the trace and use that.

High-speed, RF, or otherwise critical nets require special consideration. Directly soldering in a test point hook can compromise signal integrity, as it may introduce an electrical stub that causes reflections or other issues. A better solution is to design a test point, such as a directional coupler, that won't disturb your critical net.

Along with adding test points, break out the clock and data lines of every bus you're using into a test connector. You don't have to populate the connector, but if you ever need to read, write, or observe anything on that bus, having a connector will make the process much easier than having to solder wires to component pins. If you're using a high-speed bus, this probably won't be as simple as running another set of wires out

to a connector, since you might incur reflections or other issues that can prevent the bus from working correctly in the first place. In those special cases, follow the guidelines in the standard of the bus you're using.

If you have extra GPIO pins on a microcontroller, stick some LEDs on them. You don't need to populate the LED, but it'll be useful for providing feedback during testing and debugging (and it gives you more flexibility for a rework if you decide you need to add another input or output). During your first "Hello, world!" test, for example, the LED can indicate that the microcontroller was programmed correctly and has control of at least one GPIO pin. Similarly, during embedded software development, the LED can signal when you've hit an interrupt or performed a task. You can also put an LED on each power supply to indicate when it's on and functional. Some power-supply ICs have a dedicated "power good" pin for this purpose. Likewise, placing an LED on a data bus provides visual verification that bits are moving. Just make sure to set the LED up so it's at the right voltage and doesn't interfere with the bus.

Measuring power consumption during bring-up is always important. You can usually accomplish this by powering your board with a bench power supply and watching the current draw. If you want to measure current at specific points on your board, try including a very small value current-sense resistor in series with the power supply. This will make it easy to measure with a voltmeter and then calculate the current. You can also make the resistor 0 Ω and not populate it, allowing you to connect an ammeter in series with the trace of interest. In either case, when you're done you can just put in a 0 Ω resistor (for example, during production).

If you have an SPI bus in your design (or any bus that has a chip select), make sure to put pull-up or pull-down resistors on the chip-select pins of each device on the bus to ensure that they're disabled (or unselected) by default. This will make debugging much easier, since you won't have more than one device vying for control over the bus. It's best to use external resistors for this rather than relying on internal pull-ups or pull-downs on your microcontroller. This way, you'll avoid bus

contention even if you can't set the registers in your microcontroller to enable the pull-up or pull-down resistors. Most SPI devices are set up to require a pull-up resistor on the chip-select line, since that pin is usually active low.

If your design incorporates a UART, you can connect the TX line to an interrupt on the microcontroller so that when communication starts, the microcontroller will wake up if it's asleep. If you have a dedicated programming UART, you can use that interrupt to enter your firmware's debug mode when you start a serial terminal session.

It's sometimes useful to put both a pull-up and a pull-down resistor on a reset line so you can choose if you want the device to be default enabled or disabled. This can make debugging easier, since you can put everything in reset except for the part that you're working on. But make sure to mark one of those resistors as DNP! See "Review Checklist" on page 92 for more.

Adding an in-circuit test mode can be very helpful during PCB bring-up and verification and in full production. The details of designing a test mode and preparing it for full production are outside the scope of this book, but in general, any test modes should be able to exercise all device functionality and allow the board to be verified after it's been assembled.

In this section, we've talked about using 0 Ω resistors as a way to make your design more flexible and easier to debug, but it's important to think about whether your 0 Ω resistor will actually look like a short to your signal. For a digital signal, rise time is really what matters. Many modern digital chips have a rise time of 1 ns or faster, and running that through a 0 Ω resistor may slow down the rise time or affect the waveform because of the resistor's inductance. Exactly how much inductance a resistor has depends not only on the package size but also on factors like the termination style, trim style, and how it's mounted. In general, a physically smaller package will result in less inductance. Using a 0 Ω resistor is often totally fine, but remember to pay attention to the signal you're putting through it, and consider the non-ideal effects the resistor may cause.

Ensuring Performance

The best way to ensure your design performs well is to read the application notes and datasheets for your components and follow what they say. Application notes are written for your benefit. Are they always the pinnacle of technical and pedagogical perfection? No, but you can probably learn something from every application note anyway, and often you'll find information that doesn't appear anywhere else. The component designers have already spent time figuring out how to get the best performance out of their devices—that's how they can put those really nice numbers on the front of the datasheet to convince you to use their part. Following the manufacturer's schematic and layout recommendations gives you the best chance of actually achieving the advertised performance.

That said, the official recommendations aren't necessarily the be-all and end-all. If you know what you're doing, you can make changes that will improve performance for your application. Here are some factors to consider and tips that can help:

Current limits Check the current limits on any load switches or power supplies. Use your power budget to figure out what the maximum current (and power) through any power supplies will be. Some load switches will further constrain current. Your power supply may be rated to provide enough power, but if your load switch has a lower limit, your device still won't work.

Clock buffers If you're driving multiple loads with a single clock, crystal oscillator, resonator, or RTC, use a clock buffer. If you don't, you may exceed the fan-out capability of your output device. This can reduce the signal's amplitude to below your device's logic thresholds. For example, this may occur if you chain together a large number of I2C or SPI devices. Your data lines may also need to be buffered, since a large number of devices on the same wire will present a large combined input capacitance that may make it difficult to drive.

Power dissipation Make sure that the power dissipated in all components is below their thermal and power ratings. On parts like amplifiers, you'll often need a heat sink when the power dissipated will be greater than the power that the package is rated for. Parts are often available in several different packages, so choose the one rated for the power you need. In general, the larger the part, the more heat it can dissipate.

Load switches To reduce power consumption, use a load switch to switch out sections of your circuit when they're inactive. You may be tempted to save money by using a transistor instead of a load switch, but it may have a higher leakage current and it won't include ESD protection.

Plans for all states Use a flowchart or UML diagram to map out the state of all devices at any given time. Then make sure you aren't counting on some functionality that won't be available when a device is in a particular state. For example, devices like microcontrollers disable some features when they're in low-power modes, so you can't always rely on those features.

Soft starts If your SMPS has a soft start option, use it. The soft start will slow the output voltage's rise time as the power supply stabilizes. This generates less harmonic noise at startup and doesn't stress the battery or SMPS input voltage source by drawing too much current too fast. It can also help reduce inductive flyback.

Maximum ratings Don't use parts at their maximum ratings, including the upper end of their tolerances, and the maximum voltage, current, speed, and so on. This will prolong the life of your components, and frankly you're just asking for trouble if you use parts that are *barely* within their rated range.

Certain components require special consideration, as we'll discuss next.

Capacitors for ICs

All ICs, especially digital ICs, need multiple decoupling/bypass capacitors. Larger-value capacitors have a low impedance at low frequencies, while smaller-value capacitors have low impedance at higher frequencies. Using multiple capacitors of different values in parallel helps maintain a low impedance to the power supply across a wide range of frequencies. For example, while a 10 µF capacitor in parallel with a 1 µF capacitor is theoretically equivalent to a single 11 µF capacitor, the resulting performance is actually completely different. As the larger capacitor starts to increase impedance, the smaller capacitor starts to take over.

Most datasheets and application notes will tell you the recommended bypass capacitors to use (how many, what values, and where to put them). These recommendations are usually enough to get the circuit to work, but there's a good chance that the design will fail EMI/EMC testing. An excellent book on how to design your power distribution network correctly is *Principles of Power Integrity for PDN Design, Simplified* by Eric Bogatin and Larry D. Smith (Pearson, 2017). I recommend picking up a copy and following the advice in there.

The chemistry and physical size of the capacitors also matters, so pay attention to the package size, capacitance, and ESR recommendations of the datasheet and application notes. Physically smaller capacitors have less inductance, for example. See "Capacitors" on page 27 for more information.

Ferrite Beads

You must take special care selecting and placing ferrite beads. If you use one to help reduce power supply noise, for example, it should be placed in front of the bypass caps (that is, current should flow through the bead *before* the bypass caps). If the bypass capacitor goes before the ferrite bead, the IC will be unable to draw energy for current spikes from the capacitor because the inductive part of the bead will oppose any changes in current. You might as well not even have a capacitor there, and your design will do more harm than good. However, don't use a ferrite bead unless you have a good reason to, and before you add it to your design,

test your design with and without the bead and determine if it's actually helping.

Placing a ferrite bead on the output of a switching power supply is a good way to prevent noise created by the switching from getting to the rest of the circuit. Most SMPS datasheets will advise you on what bead to use and where to put it, but generally you want the bead on the output with one capacitor on either side of it.

Since ferrite beads are essentially lossy inductors, they can combine with nearby capacitors and form an oscillating LC tank circuit. This accidental oscillator will start spewing the following frequency into your circuit:

$$f = \frac{1}{2\pi\sqrt{LC}}$$

To avoid this, it's important to pick ferrite bead values that won't resonate with capacitors, both those you designed in on purpose and those that occur accidentally through stray or mutual capacitance. Every ferrite bead will resonate at least a little because it's impossible to eliminate all stray capacitance, but the goal is to keep this resonance from interfering with your signals of interest and prevent it from causing EMI/EMC problems.

If there's a lot of bypass capacitance next to your ferrite bead, it's a good idea to simulate that network in a tool like LTSpice to see if oscillation will occur. Use the SPICE models of the actual parts you'll be working with, if possible. Parasitic capacitance may make it difficult to accurately simulate your circuit, however, so it's also a good idea to have a kit of ferrite beads on hand during prototyping. This way, you can measure the actual oscillation and try different bead values until you find one that works well.

If you're having trouble eliminating ringing from a ferrite bead, try reducing the effective Q of your accidental oscillator by adding a resistor. Figure 5-5 shows the best way to do this without affecting any other performance characteristics.

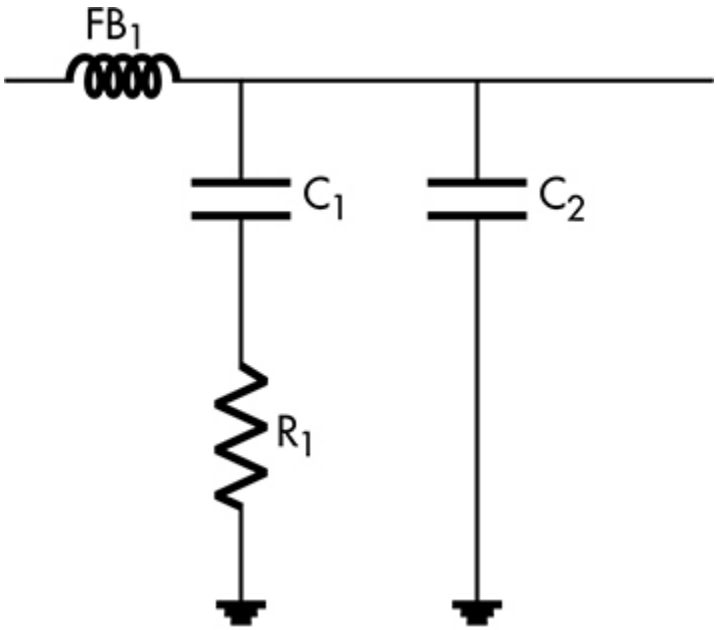


Figure 5-5: Adding a resistor to reduce the Q of the oscillator caused by the interaction of the ferrite bead and the two capacitors

R_1 is behind a DC-blocking capacitor, so there isn't a large voltage drop like there would be if it was in series with the ferrite bead, FB_1. Capacitor C_2 is a small-value bypass capacitor (something in the nanofarad range), and C_1 is a much larger capacitor (something in the microfarad range). This reduces the Q of the resonator without significantly affecting the high-frequency bypass performance of the capacitors. R_1 should be small, around 1 to 3 Ω. You can try simulating it, or use the foolproof method of testing: soldering in the parts and measuring it.

Amplifiers

Especially in RF design, isolation between amplifiers is important to prevent unintended oscillation. One way to do this is to put a pi network between each amplifier stage and use it to add some attenuation. Oscillation is caused by some of the output getting back into the input, so attenuating the input can help reduce that unwanted output signal below the critical threshold that causes oscillation.

You should also be careful not to put too big a capacitive load on your amplifier outputs. This applies to all amplifiers, not just RF parts. As the capacitance increases, so will the phase shift from the output to the input. The closer this loop phase shift gets to 180 degrees, the more likely it is that the amplifier will oscillate. However, a capacitive load isn't always the result of a physical capacitor. It can also be caused by driving a long cable or another part with a large input capacitance. The best way to avoid this problem is to stay below the maximum drive capacitance listed in the amplifier's datasheet.

You might also find a graph of small signal overshoot versus capacitive load in the datasheet, and you'll want to make sure you choose a capacitive load that keeps the small signal overshoot low. Once small signal overshoot reaches 100 percent, congratulations, you've just built an oscillator. If you're designing your own amplifier, the maximum capacitive load you can drive will depend on your gain. It's possible to design an amplifier that can drive *any* output capacitance and not oscillate, something known as an unconditionally stable design. It's especially important to design input and output stages to be unconditionally stable.

Conclusion

In this chapter, we covered the importance of a schematic as the primary document indicating your design intent. You learned about conventions and best practices for producing clear schematics and about common mistakes to avoid. You also learned how the right choices in your schematic can facilitate debugging and optimize your design's performance. If you've followed the advice in this chapter, when some future engineer you've never met is trying to figure out how to replace a component in your design or create something new that needs to be backward compatible with your device, they'll deeply appreciate how much easier you've made it thanks to your thoughtful schematic.

6
LAYOUT DESIGN

Circuit board layout can be hard to get right. You have a schematic that shows all of the electrical connections you need to make in your layout, but if you naively make all of those connections, your board probably won't work. Instead, the layout step requires careful thinking about the practical implications of what you're trying to do. It actually matters where components go in relation to each other, what material you make your PCB from, what size your traces are, where vias go, and more. In this chapter, we'll cover the myriad of questions you need to consider when laying out your board so that your design actually works.

Keys to Success

What makes a layout successful? Good layout is driven by good placement. Good routing should naturally fall out of your placement choices. Beginner designers will often place parts in a haphazard way and spend the majority of layout time routing traces. Instead, you should spend more time thinking about where to place parts, what rotation they should have, which side of the board they should go on, and the like. This can't be overstated: Good placement is absolutely critical to designing a PCB that works. If you place parts without

thinking about the fields and currents of your design, you'll encounter problems with your power distribution network, signal integrity, electromagnetic compatibility, and even manufacturability.

When placing components, parts *must* be clumped together by function. For example, your analog circuitry should be confined to one area, your digital circuitry to a different area, your power supplies to a third area, and so on. If you don't do this, your board will be very hard to route, and different parts of your design will interfere with each other.

To help make this happen, the first thing you should do after you finish the schematic is arrange the parts into groups. Then you can take a first pass at placement and begin seeing where components will have to go on your board outline by dragging around your roughly placed groups.

Grouping parts by function like this is required for a couple of reasons. First, it will help prevent crosstalk and noise from coupling into different subsystems. Second, it will make troubleshooting, board bring-up, and testing easier, since you can isolate, modify, and replace parts in a single area instead of all over the board. It also makes routing much cleaner. It's almost impossible to correctly lay out a PCB if you spread parts from all systems all over the board.

A good design is also maintainable and repairable. Even though your pick-and-place machine can place parts with microscopic precision, think about how hard it will be to replace components by hand if something needs to be fixed. How good do your tolerances need to be? Can you fit a soldering iron in there? How many screws do you have to remove? Do wires need to be desoldered to get at anything? If you assemble your engineering prototypes by hand, you'll quickly discover any shortcomings in maintainability and repairability. Sometimes it's impractical to assemble your engineering prototypes by hand, or you'll need a technician to do it. In that case, talk to your technician. Ask them what's frustrating to repair or rework so you can incorporate their feedback into your next revision. In general, be careful about placing components too close together or in such a way that they will interfere with tools that need to be used on the PCB.

There are also mechanical placement requirements to think about. Besides obvious factors like component height, connector overhang, and mounting and screw holes, don't forget about more subtle considerations like how far the leads on the bottom of through-hole parts stick out. One good way to detect and avoid these problems is to use a CAD package with 3D modeling support. Most electrical CAD programs support 3D component models and even allow you to export to a mechanical CAD program (like SolidWorks or Catia) so your mechanical engineer or industrial designer can do a fit check before fabrication. Using this technique requires accurate and complete 3D models of all components, but many manufacturers provide 3D models of their components you can download for free.

Another key to a successful design is to have other people review your schematics and layout before you send the PCB out for fabrication. Try to involve people from across the production process: other electrical engineers, production engineers, mechanical engineers, test engineers, assemblers, and so on. Having experts from different fields look over your design will help you catch issues that you may have missed from your perspective but that will be apparent to people in other disciplines. You need to understand the needs of your whole team, not just the end user of the product. PCB fabricators and assemblers, in particular, often have a tremendous amount of practical knowledge from experience that can dramatically improve your DFM and prevent major problems or delays you otherwise wouldn't have known about. Don't try to build a design without talking to them! I recommend talking to PCB fabricators and assemblers early, even before you hold a cross-functional design review.

It's best to have the reviewers look the design over on their own before you hold a formal design review meeting with the group. In the meantime, the reviewers shouldn't consult with each other or share what they've found. The more independent the reviews, the more likely you are to catch everything. For the same reason, it's dangerous to divide the design up into chunks and have each person look at a different piece. You want as much redundant coverage as possible. Allow everyone at least a day or two to evaluate the design before the review

meeting. While you're waiting, freeze the design, or else you may end up making a change that slips through without being reviewed.

When I review designs, I like to print them out on a piece of paper, go to a quiet room by myself with a stack of datasheets for every part used in the design, and work through the whole thing again. It's still useful to use your CAD software to inspect layout features that may be difficult to see on a printout, but for me, looking at everything on paper, with zero distractions, helps me pay attention to detail. You might encourage your reviewers to do the same.

While feedback is invaluable, don't mistake a design review for a design committee. You own your design. You should take responsibility for your errors, but you should also make the final call on what stays in the design and what goes. You won't be working in a vacuum, of course, and you should absolutely listen to and take the advice of other engineers (especially those in disciplines in which you don't have as much experience), but you need to have a good reason for every decision you make. You're the voice of your product during development, so stand up for it. If you know you're right, don't give in. If someone requests a change that you can clearly demonstrate will be bad for the product, refuse the change. Ten people can't all design one PCB. Ten people may work on it, but there needs to be one design lead who has the final say.

Designing for Performance

Two different people can lay out the same schematic and end up with PCBs that perform quite differently from each other. Designing for performance means thinking about the physics of fields and waves as you manipulate them in your design. If you don't account for how signals behave in real life, your design won't act like you expect.

PCB Physics

Many engineers have an incorrect and oversimplified mental model of PCB physics. They imagine that different components need different voltages and currents, and that you use traces, which are just like wires,

to supply voltages and move currents. Current flows into components and then terminates when it goes back to ground.

Here's how things actually work. First, everything acts as an electrical component, including the traces and the PCB itself. These phantom components are often called *parasitic inductance* and *parasitic capacitance*. There's no such thing as "just a wire." The currents and voltages on traces have an associated electromagnetic field, and those fields aren't contained within traces like water in a pipe: They surround the trace. This means there are fields in between your PCB layers. Think about those fields for a moment and you can begin to get some intuition for how crosstalk can happen or how noise from one circuit can spread to another.

The way current flows through a trace changes based on the frequency. With DC, current flows through the entire cross section of the trace, and current is inhibited only by resistance. However, at high frequencies, it's a different story. Only a small part of the cross section of the trace, just the outer "skin," carries the current. Resistance still inhibits current flow, but inductance acts as a much larger inhibitor.

Ground isn't a current sink. Current doesn't magically terminate when it hits something you've called "ground" in your schematic. The energy that powers your components is actually contained within the electromagnetic fields that surround the trace. On a PCB that has a trace on the top layer and a ground plane underneath it, the path that current will take through the ground plane at steady state depends on its frequency. Here are two possibilities:

- Since DC current is inhibited only by resistance, it will follow the path of least resistance on its return path. Usually this means taking the shortest direct-line path between the current source and the current load.
- At higher frequencies, current is inhibited mostly by inductance, so the return current will follow the path of least inductance. This means the return current will follow a path directly underneath the trace.

You can see three example scenarios in the simulations in Figure 6-1.

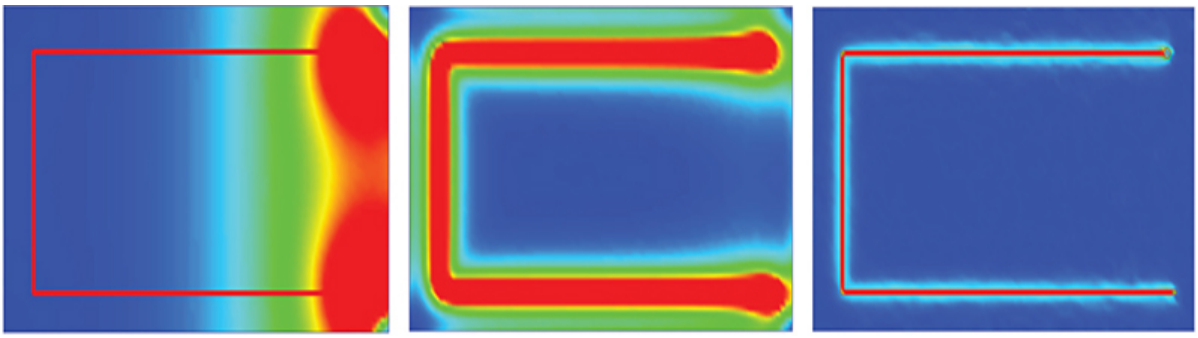


Figure 6-1: Current density at 1 kHz (left), 1 MHz (center), and 100 MHz (right)

All three images show a trace that starts at the bottom right, jogs to the left, goes up, and then jogs back to the right. At steady state, current flows from one end of the trace to the other, and there's a solid ground plane beneath the trace. The plot on the left shows current density when the frequency of the signal through the trace is 1 kHz. The center plot shows current density when the frequency of the signal through the trace is 1 MHz. The plot on the right shows current density when the frequency of the signal through the trace is 100 MHz. We're looking at all current in these images, including return current.

Take a close look at the current flowing in the vertical gap between the top and bottom of the trace on the right side. In the left plot, even though there's no trace there, current is still flowing. This is the return current flowing through the ground plane, and it's taking the shortest straight-line path between the beginning and end of the trace, since that's the path of least resistance. In the plot on the right, there's no return current flowing between the two ends of the trace. Instead, it's all returning back underneath the trace, since that's the path of least inductance.

These plots also illustrate how fields surround a trace, occupying space in the air above the trace and in the PCB substrate beneath it. This is why it's so important to use ground planes: If you don't use a ground plane, you're constraining where the return current can flow. This will create large loops of current, which will cause EMC and EMI

problems. The other reason it's so important to use ground planes is that fields can't penetrate them. When you put a ground plane between each pair of signal layers, the fields from those signal layers will be blocked by the ground plane and thus be unable to couple with each other or cause interference problems.

It's still possible to design PCBs that work even if you're operating off an incomplete mental model of PCB physics. In particular, for designs that aren't high speed or that are relatively simple, you can get away with not following these rules. Still, it's greatly beneficial to start designing with a good physical understanding of what's really going on, since you'll encounter fewer problems with your designs and actually stand a chance of passing regulatory testing.

NOTE

If you'd like to learn more about PCB physics, check out the book Fast Circuit Boards: Energy Management *by Ralph Morrison (Wiley, 2018).*

Design Tips

The following tips will help you lay out your design with the board's performance in mind:

Always include a ground plane. This will prevent a myriad of problems and significantly simplify routing. Instead of drawing a trace back to your ground net, you can simply drop a via. Where and how big of a ground plane you use will depend on your design limitations. Ideally, a two-layer PCB will have a ground plane on the bottom side and everything else on the top. If this isn't feasible, at least keep ground planes under critical ICs and nets. Resist the temptation to flood-fill a ground plane and then start running traces through it when you run out of room on the top layer. This will effectively create holes in the ground plane copper that can cause EMC problems.

Don't share ground vias on capacitors. This is especially critical in RF and high-speed circuits. Having a single via to ground shared by multiple bypass capacitors adds unnecessary inductance and reduces the performance of the bypass capacitors by increasing impedance at high frequencies. Extra vias don't cost you anything, so use them liberally.

Place feedback resistors as close to their IC as possible. This is important because at high frequencies, longer traces between the IC and the resistor will add stray inductance and can change the performance of your circuit. Longer traces can also act as loop antennas, inductively or capacitively picking up noise or crosstalk from elsewhere on the PCB.

Don't flex sensitive components. A resistor's value actually changes when it's flexed, so if the resistor is high tolerance, the simple act of flexing a PCB can throw the resistor value out of spec. If your design has high-tolerance resistors, check to make sure they won't get flexed when mounted or during normal use. Mechanical stress can also affect capacitors, voltage references, and small CSP chips. Don't place flex-sensitive components next to mounting holes, slots, or edges in the PCB.

Place nearby inductors at right angles to each other. An inductor or coil creates a magnetic field that goes through the middle of the windings, and that field can interfere with the fields of other inductors if they're too close. Putting the inductors at right angles keeps them from coupling to one another, since their fields will be perpendicular. Other good ways to reduce coupling between inductors are to keep them far apart and to use shielded inductors.

Remove solder mask with care. If you decide to remove solder mask from parts of your PCB that contain ICs, try leaving a few mils of solder mask around the component footprint(s). This will help prevent solder paste from wicking off the pads and onto the surrounding exposed copper. For example, in Figure 6-2, solder mask has been pulled back from over top of a coplanar waveguide but not all the way up to the pin of the chip.

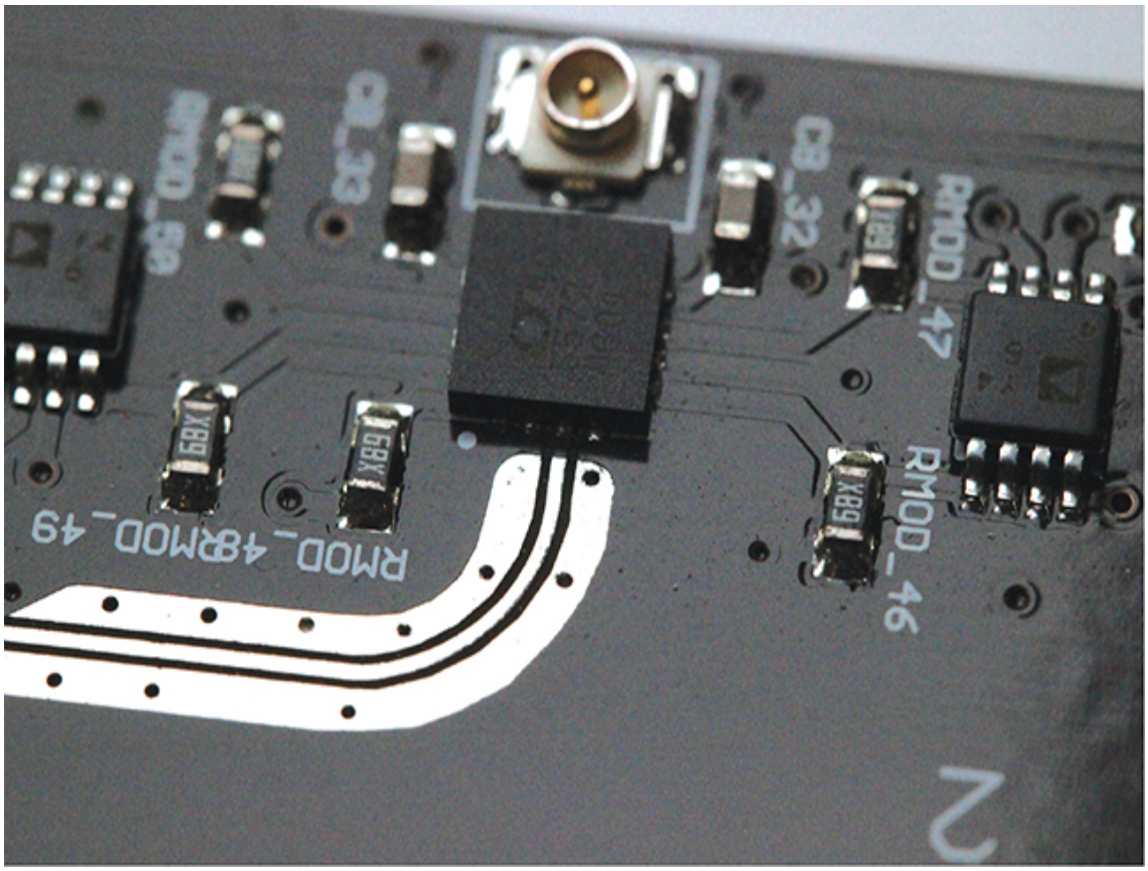


Figure 6-2: A small amount of solder mask prevents the IC pad's solder paste from wicking down the coplanar waveguide that it's feeding.

Be careful under SMPSs. Don't route anything under an SMPS besides traces for that power supply. A possible exception to this rule is if there's a full ground plane between the SMPS and your non-SMPS traces. SMPSs create noise as a result of the switching FET. If the FET switching looks roughly like a square wave (and it often does), it will also generate higher-frequency noise from those waveform edges. The node where this switching happens is sometimes called the LX node. Running traces under the power supply can couple that switching noise into those traces. You should therefore place SMPSs away from sensitive parts of your design. Similarly, you shouldn't route anything underneath other sensitive circuits unless they're separated by a solid ground plane.

Don't route digital and analog lines close together. The lines can couple noise and transients into each other, especially if the rise

time of the digital signal is fast. You'll most commonly encounter analog/digital coexistence issues when designing with ADCs. Low-impedance analog lines are less affected by coupling problems because the current that couples in produces only a small change in voltage across a low impedance. On high-impedance analog lines, the same small noise current produces a higher voltage across a higher impedance. Be sure to follow the layout guidelines in the datasheet, especially for fast or high-resolution ADCs.

Don't separate analog and digital ground planes. For that matter, don't split or separate any ground plane. Some books and application notes will recommend splitting analog and digital ground planes to stop noise from the digital circuits from contaminating sensitive analog circuits. The reality is that if you physically separate your analog and digital circuits, use correct layout techniques, and use the right stackup, you can have analog and digital circuits on the same ground plane without a split and not have noise problems. You're much more likely to have terrible EMI problems from a split ground plane. For more information, see Chapter 8.

Use traces to power analog circuitry. Routing power with traces is usually better than using a power plane that sits underneath everything. Sensitive analog lines may capacitively couple to the power plane, which is probably going to contain voltage ripple. Instead, choose a stackup that puts a ground plane directly under your analog parts, and then route individual traces to each chip for power.

Route clock and data lines together. This gets more important as the communication bus's data rate (and therefore the clock rate) increases. Eventually, if the clock and corresponding data lines have significantly different lengths, there can be phase offset problems, and your clock and data edges won't align correctly at the receiver.

Mind the trace widths. Any traces that carry high current need to be wide enough to carry that current without dropping the voltage significantly or heating up. There are lots of trace width calculators

available online. In some cases, a pin on an IC that needs to carry a lot of current may be smaller than the width of the trace needed to safely carry that amount of current. The best way to address this is to increase the width of the trace as quickly as possible to "neck it up." If there are any layer changes along your high-current paths, you'll also need to use multiple vias at each transition to ensure low loss.

Mind the trace weights. Another factor to consider for high-current traces is the weight of copper to use. A trace made of 1-ounce copper will carry less current than the same trace made of 2-ounce copper. For a more complete explanation of copper weight, see "Stackup Design" on page 121.

Minimize RF layer transitions. Don't use lots of signal vias with high-speed and RF traces. Vias in a transmission line cause reflections, which increase the loss and decrease the power through the trace. Vias can be specially designed into a transmission line to minimize this loss, but avoid them if at all possible.

I encourage you to take the time to build your understanding and intuition of how current flows and how fields behave. You'll have a better understanding for why design rules like these exist than if you just memorize them.

Testing Considerations

Test points are free, so use them freely in your layout! Place them on the input and output of any power supplies, data lines, clock lines, enable lines, and the like. If a chip has a "power good" or other status output that you're not using, route it out to a test point anyway. This will help when debugging and also makes reworking easier later if you decide to use that signal for something.

You can use an exposed circle of copper as your test point, or you can use a test point that gets soldered on to make it easier to clip into. The Keystone 5018 and 5029 are great surface-mount test points, and the Keystone 5001 is a great through-hole test point. Small copper test

pads are often used for automatic testing fixtures, while larger copper test pads or soldered test points are often used for testing by engineers.

If your PCB has components on only one side, consider placing test points on the back (the side without components). This makes building a test fixture easier, since you can flip the board upside down and use a flying-lead tester, a bed-of-nails jig, or handheld probes without having to worry about components getting in the way. It's easiest to support the PCB along its edges, so make sure to leave clearance along the edges of the board. Even if you have components on both sides of the board, put all of the test points on the same side of the board if you're going to be using a bed-of-nails test jig. It's significantly more expensive to build a bed-of-nails jig that probes test points on both the top and the bottom of a PCB than it is to build one that probes test points on a single side.

Test points should be at least 30 mils in diameter, but a diameter of 40 mils is ideal. If you want to use via pads as test pads, set the via annular ring to be 30 or 40 mils in diameter.

When placing test points, it's also helpful to place a small ground pad next to the test point. This is really nice when you're probing signals on your test points because you typically need a ground connection. Placing that ground connection close by reduces stray inductance and prevents your probe from acting like a loop antenna. Most oscilloscope probes have a small spring-like attachment designed to be used exactly for this purpose. Instead of scraping off solder mask when you use it, you'll have an easy ground pad right there.

Make sure the test points are spaced far enough apart that they can all be accessed at the same time by a technician or a test fixture. When deciding which signals to make available on a test point, think about your test plan (more information on that in Chapter 13). Ensure that all of the signals you'll need to test are easily accessible.

Besides building test points into your design, request that the company that makes your PCBs includes a test coupon, especially if your design has any impedance-controlled signals. A *test coupon* is a PCB that has alignment features, copper layers, and, usually, a transmission line or differential pair on it that allows you to verify that the stackup they used was correct and that the impedance was controlled well

enough. This is useful for single-panel production runs as well as for mass production, when you want to know that a panel has a manufacturing defect before you waste time and money populating and testing those boards.

If you simply request a test coupon, the PCB fabricator will use a standard test coupon design that they have. Alternatively, you can design your own test coupon if there are particular features or specifications you want to verify. Along with a test coupon, you can also add a test stack to the edge of your PCB panel. A test stack has a pyramid-type structure of the stackup that you can use for sectioning and analysis to be sure the fabricator hit the tolerances you expected and that the stackup is correct.

Another thing you can add to a PCB panel is a bad-board marker. This is a small silkscreen square by each PCB in the panel that you can mark with a permanent marker to denote that a particular board is bad. This is useful when you populate a panel and test each board before separating individual PCBs from the panel. If any particular board fails the test, you can mark that it's bad so that when the boards are separated, the ones that failed the test can go back to be reworked.

Creepage and Clearance Requirements

If your design must adhere to a standard (especially a medical standard), you'll encounter creepage and clearance requirements. *Creepage distance* is "as the snail crawls." In other words, if you were to put the tip of a pencil on one object and move it without lifting the tip to another object, passing through holes and connectors and going around other components, that would be the creepage distance. The *clearance distance* is "as the crow flies." In other words, if you use a pair of calipers to measure the shortest distance between two objects through the air, that's the clearance distance.

Creepage and clearance requirements are designed to prevent arcing, which can occur through several different mechanisms. First, dust can be electrostatically attracted to traces at high voltage. This dust can harbor moisture and cause either an arc or a weak short. Another way arcing happens is through dielectric breakdown of the air or

whatever material is separating two points with a high potential difference. If you follow the IPC or IEC standards for creepage and clearance, you won't have to worry about arcing.

High-Speed Design Considerations

Throughout this book, and the literature in general, you'll see references to high-speed designs, but what actually qualifies as "high-speed?" Signals above 1 GHz certainly seem to count, and I've even heard people refer to a 3 MHz signal as high-speed. In reality, for signal integrity purposes, what matters are the rise time of the signal and the length of the trace it's traveling down. A good rule of thumb is that a trace whose propagation time exceeds one-quarter of the signal rise time will start showing high-speed effects. To figure out propagation time, you can use online calculators or you can simulate it, but for a quick estimate, assume that your signal will propagate at about 150 ps/in (or 59 ps/cm). Let's look at an example. If your signal has a rise time of 1 ns (1,000 ps), it will start to show high-speed effects when on a trace longer than about 1.5 inches (38 mm). This is because one-quarter of the rise time of 1,000 ps is 250 ps, and 250 ps divided by 150 ps/in is 1.6 inches.

To understand why rise time and trace length are most important, consider a square wave. To make a perfect square wave, you need infinite odd harmonics of sine waves. As you remove more and more of the high-frequency odd harmonics, the straight edge of the square wave begins to droop, and the rise time gets lower and lower. A signal that may have a low fundamental frequency can still have high-frequency components if the rise time of the edges of the signal are fast.

For most applications, an extremely fast rise time isn't necessary. In fact, unnecessarily fast rise times are one of the major causes of EMI problems (see Chapter 8 for more information). If your signal or clock frequency is below about 50 MHz, you can reduce the rise time by simply adding a series resistor. Start with something like a 60 Ω resistor and check your new rise time on an oscilloscope. The longer the trace, the higher the resistor value will need to be. For signals and clocks that

are faster than 50 MHz, a series resistor can cause timing problems. Instead, you need to treat those faster signals as high-speed.

NOTE

When in doubt, it's a good idea to drop a series resistor footprint on signals you suspect may need to have their edges slowed down. If you don't end up needing it, you can just populate a 0 Ω resistor.

Remember that when you're calculating the wavelength, you need to take into account the medium in which the signal is propagating. Real PCB substrates have a velocity factor. For example, in FR-4, electromagnetic waves will propagate about 50 percent slower than in free space, which will make the effective wavelength shorter. This is especially important to keep in mind when considering whether a trace or feature is a significant fraction of the signal's wavelength. To calculate wavelength in a medium (in millimeters), use the following equation:

$$\lambda = \frac{300}{f\sqrt{\epsilon_r}}$$

Here, f is the frequency in GHz and ε_r is relative permittivity.

Don't simply split or fork high-speed signal traces. This includes clocks, data lines, and RF traces that are running above 10 MHz. Splitting these signals without special consideration can cause reflections that will degrade performance (or, in the worst case, destroy the driving circuitry). Additionally, many clock and data drivers simply can't drive more than a single output. If you need to split a high-speed signal, use a buffer. If you need to split an RF signal, use an RF splitter. Make sure the component you choose is rated for the frequency at which you're using it.

Impedance Matching

If your signal is on a trace length longer than one-fifth of the signal's wavelength in the PCB dielectric, you'll need to do impedance matching. For example, a 1 GHz signal has a wavelength in FR-4 of

about 5.9 inches (150 mm). That means that any trace longer than 1.18 inches (30 mm) will need to be impedance controlled. To do this in a digital circuit, that line will need to be resistor-terminated at either the driver or the last load on the line.

There are two ways to terminate a digital line: series or parallel. To series-terminate the line, you need to determine the output impedance of the driver and then pick a series resistor that, when added to the IC output impedance, equals the line impedance you intend to create. For example, if you intend to create a 50 Ω line and the output impedance of the driver is 15 Ω, you'll need a series resistor of approximately 35 Ω (35 + 15 = 50). The series resistor will need to be placed very close to the output of the driver IC. For a parallel termination of the line, you would simply place a resistor of about 50 Ω very close to the last load of the line, between the load's input and ground. You should prefer to use series termination, since it's far superior at lowering EMI and crosstalk.

To impedance-match an RF trace, the trace impedance should simply be designed with the right dimensions to match both the driver's output impedance and the load's input impedance. No resistor is needed.

A trace's impedance depends on many of the PCB's physical properties, such as the trace height and width, the PCB material, the number of layers, and the distance between those layers and the trace. There are calculators online that can help you plan out the dimensions needed to arrive at your correct final impedance. These typically work in two directions: You can type in the dimensions that you have and the calculator will tell you the resulting impedance, or you can give it your desired impedance and it will calculate the trace width required to get there. Some ECAD programs have built-in tools to calculate impedance, or you can use simulation tools like Keysight ADS to model your PCB, including vias and transmission lines. You can also use open source tools like OpenEMS or Elmer.

If your design requires any traces that have a particular impedance, make sure to tell your PCB fabricator to use impedance control on those traces and listen to their feedback. You'll first calculate the correct transmission line dimensions using a software tool like LineCalc (made

by Keysight) or one of many free online impedance calculators (you can find links to these calculators on this book's website). However, these results (especially the ones from online calculators) will be estimates. The fabrication process will have some variance on the width, plating thickness, and spacing. Specifying impedance control will tell the manufacturer to take those variances into account and precisely hit a target impedance. They'll give you their own calculated trace widths to hit the impedance you requested, and you should use those numbers in your layout. You need to use the exact numbers they give you, since they'll use software that pulls in all of the nets with their calculated widths to track which nets are impedance controlled. You should also call out impedance control in your fab notes. The manufacturer will also continue to monitor the factors that influence impedance during board fabrication and make small changes if necessary to hit your requested value.

Don't get too concerned with designing extremely tight-tolerance trace dimensions. The standard tolerance for impedance of digital PCBs with thin (4-mil to 8-mil) lines is +/– 10 percent. You should make your design capable of functioning with this amount of impedance error. You can request tighter tolerances, but it will be very expensive. With RF boards, traces are usually wider (for example, 25 mils), are above a ground plane, and have a fairly thick dielectric (often 20 mils) between the two layers. This allows RF boards to hold a much tighter tolerance compared to digital boards but without the extra cost.

If you're feeding an antenna, add a small pi network at the point closest to the antenna. This makes it easy to add a matching network if your antenna doesn't look like exactly 50 Ω. Pi networks are also easy to bypass, since you just need to add a single 0 Ω resistor to the series component. When you're laying out your matching pi networks, make sure to keep the ground vias as close to the component pads as possible. Placing them farther away can add stray inductance that makes tuning more difficult. This is also why all of the components in the pi network should be placed as close together as possible.

Another trick you can do is add a footprint of an RF connector in line with your antenna feed and marking it as DNP. When you get your

board back, you can cut the trace right under the connector footprint and install an RF connector to face either the antenna or the RFIC to measure them with a vector network analyzer.

Signal Measurement

Slow signals can be probed by simply touching an oscilloscope, multimeter, or logic analyzer probe to test points or component pads. However, RF signals can't be reliably probed this way. Instead, you can add a U.FL connector or another similar small RF connector that's connected with a removable 0 Ω resistor. As mentioned, you can't just add a fork to a high-speed trace and expect it to still work the same way, so you need to make it easy to terminate the transmission line into a connector while disconnecting it from the rest of the circuit past the connector. A pair of 0 Ω resistors, as shown in Figure 6-3, is a good, reversible way to do this.

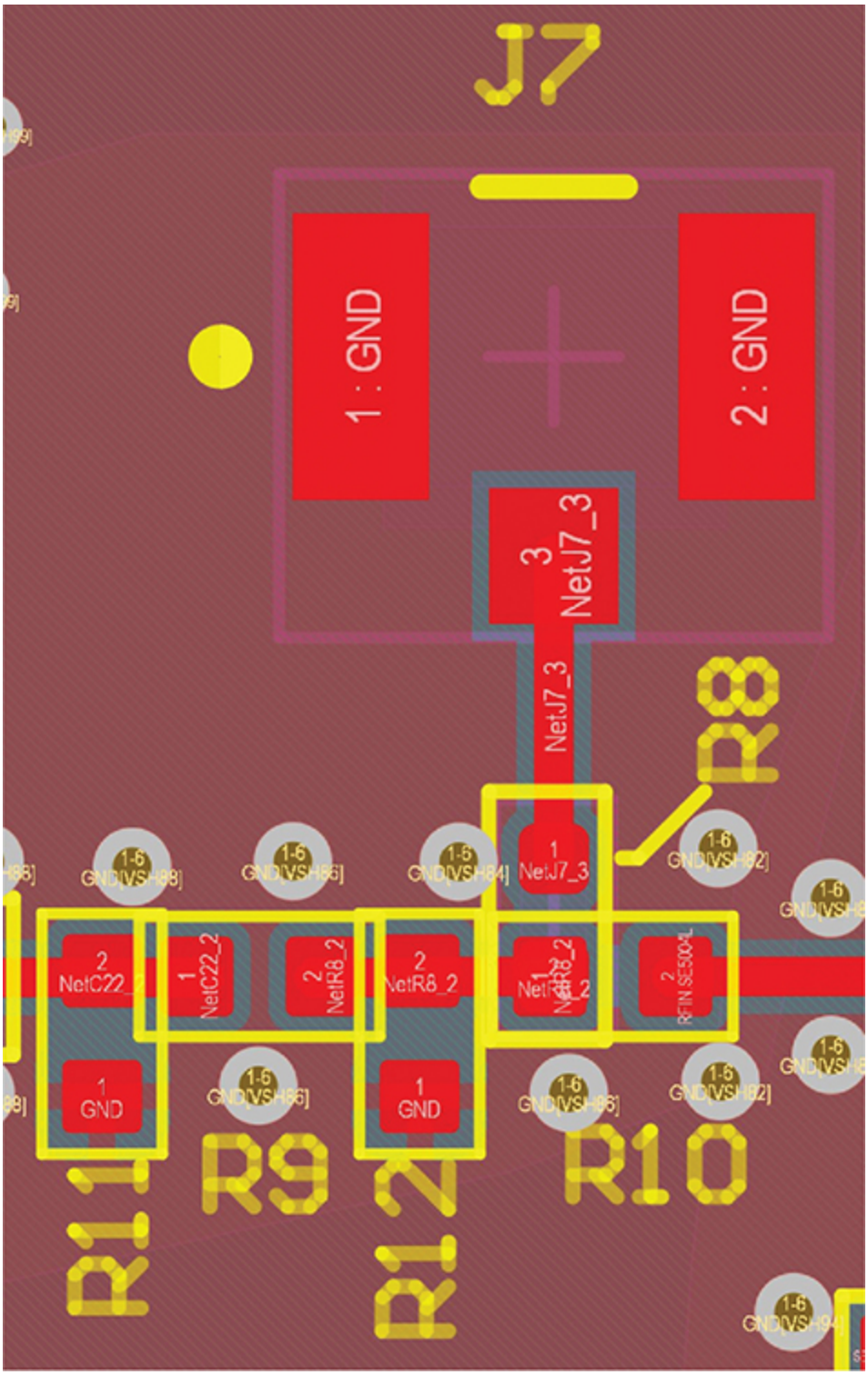


Figure 6-3: A resistor network that can be used to divert an RF signal to a connector

In this layout, two resistors are used to switch the direction of an RF signal into a U.FL connector (J7) and a coplanar waveguide. If resistor

R8 is installed, the signal is directed into the connector. If instead resistor R10 is installed, the connector is bypassed, and the signal continues along the transmission line. One potential problem with this technique is that your 0 Ω resistor will impart parasitic inductance that may affect your measurement. This will get worse at high frequencies of interest and with physically larger resistors. For simple measurements, such as determining if your signal is present at approximately the amplitude you expect, this measurement error may not matter. But if you're taking signal integrity measurements using this technique (such as for high-speed digital signals), you likely won't be seeing an accurate depiction of the signal. Instead, consider using a directional coupler as a way to safely sample an RF signal. This could be a physical part that you solder in, or you can create a directional coupler in copper. By running a trace next to your transmission line of interest, you can sample a low-amplitude version of whatever is on your transmission line.

You can also buy special oscilloscope probes that are designed to have an extremely low loading factor and can be used to measure high-frequency content. Test points for these probes can just be openings in the solder mask on the traces you need to measure. These probes can be very expensive, but they're required if you need to measure your fast signal in the time domain. Lower-cost open source probes are available.

Problems to Avoid

Beware of crosstalk between traces that are physically close to each other. For sufficient isolation between two microstrip traces, make sure the distance between them is at least four times the thickness of the dielectric between the microstrips and their reference ground planes. To prevent crosstalk or coupling between two striplines, separate them by at least double the thickness of the dielectric between the striplines and their reference ground planes. Making a little fence of ground vias between two traces is another way to keep them isolated, and coplanar waveguide transmissions lines typically already have these. Make sure that the distance between the vias is no larger than $\lambda/20$, where λ is the wavelength of the signal you're trying to isolate.

When it's not possible to have enough spacing to mitigate crosstalk because the PCB is too dense, you'll need to estimate or simulate potential crosstalk. This is an entire field of electrical engineering, so I'll point you to the authorities on the subject, who have already written some excellent material on this: Douglas Brooks and Lee Ritchey have both written articles on crosstalk due to tight spacing that are available online, and Ritchey has a book called *Right the First Time* (Speeding Edge, 2003) that speaks to this and many other high-speed design subjects.

If you have multiple radios on a single PCB, place them far apart, especially if they will be transmitting at the same time and their antennas are on the same PCB. This will help prevent interference between the radios and keep the antennas from detuning. This is called *coexistence*, and it's often more challenging than just placing radios far apart. All of the other advice in this chapter applies when trying to solve coexistence problems.

Be aware that using ENIG plating on RF traces will add a little extra loss because it contains nickel. Nickel is much more lossy than copper, and because the nickel is on the outside of the trace, the skin effect will concentrate most of the RF signal in the nickel. The result is a transmission line with more loss than you would otherwise expect. Figure 6-4 shows what an ENIG plating looks like.

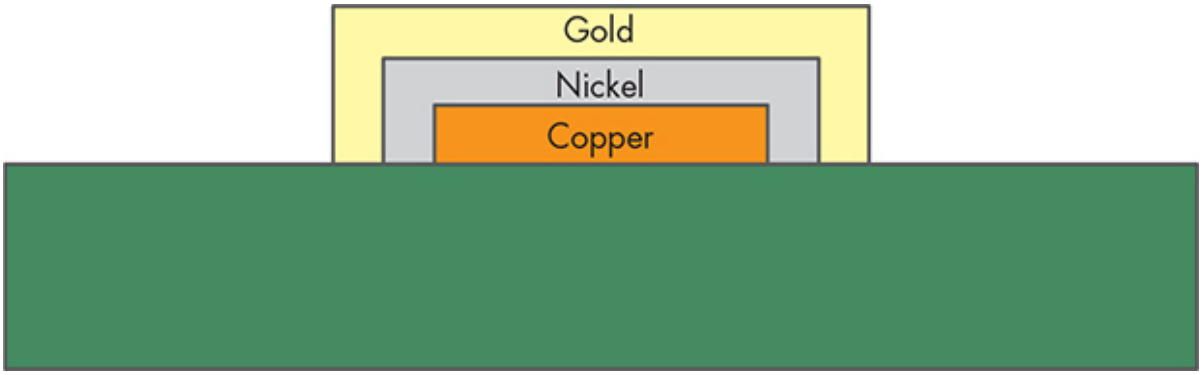


Figure 6-4: ENIG plating

Table 6-1 shows the relative loss of ENIG, measured in decibels per inch, at high frequencies for both microstrip traces and differential pairs.

Table 6-1: ENIG Insertion Loss (in dB/inch) vs. Frequency

Frequency	ENIG microstrip	ENIG differential pair
1 GHz	0.10	0.50
2 GHz	0.20	0.80
3 GHz	0.30	0.90
4 GHz	0.38	1.25
5 GHz	0.48	1.50
6 GHz	0.50	1.75
7 GHz	0.60	2.00
8 GHz	0.69	2.50
9 GHz	0.79	2.75
10 GHz	0.90	3.10

For comparison, Table 6-2 shows some examples of the loss caused by other finishes besides ENIG.

Table 6-2: Insertion Loss (in dB/inch) of Other Finishes

Frequency	Sn60Pb40 (HASL) microstrip	Bare copper microstrip	OSP microstrip	OSP differential pair
1 GHz	0.10	0.09	0.10	0.25
2 GHz	0.18	0.18	0.18	0.35
3 GHz	0.28	0.22	0.28	0.45

Frequency	Sn60Pb40 (HASL) microstrip	Bare copper microstrip	OSP microstrip	OSP differential pair
4 GHz	0.30	0.30	0.30	0.50
5 GHz	0.40	0.40	0.40	0.60
6 GHz	0.50	0.45	0.50	0.75
7 GHz	0.59	0.55	0.59	0.80
8 GHz	0.65	0.60	0.65	1.00
9 GHz	0.73	0.70	0.73	1.20
10 GHz	0.89	0.80	0.85	1.25

As you can see, the insertion loss is worse for ENIG, especially for differential pairs.

NOTE

Tables 6-1 and 6-2 were taken from Xin Wu et al., "Surface Finish Effects on High-Speed Signal Degradation," IEEE Transactions on Advanced Packaging *31.1 (2008): 182–189.*

Turns in Traces

When laying out traces, use 45-degree or mitered turns. The default trace routing style in some CAD tools is diagonal, so many novice boards have traces that run all over the board and look more like they're drawn freehand. Using 45-degree turns instead keeps things neater and helps enforce the good routing style of keeping intralayer traces roughly parallel and inter-layer traces roughly perpendicular, as shown in Figure 6-5. This is sometimes called "Manhattan-style" routing.

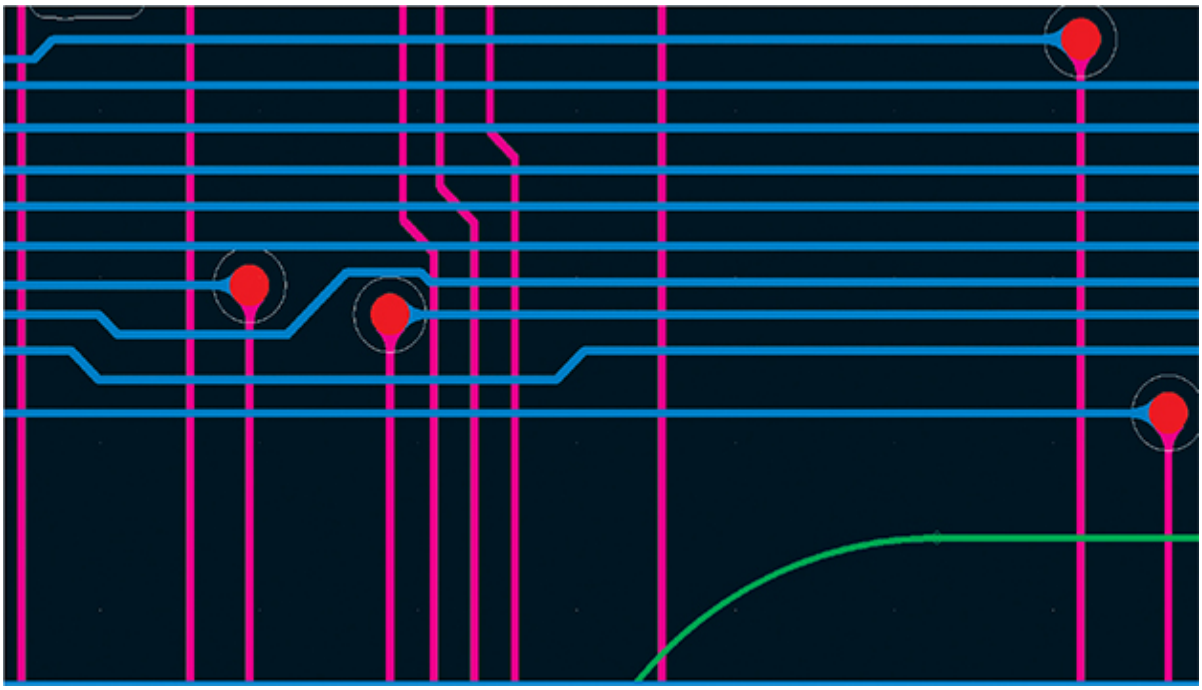


Figure 6-5: Traces on different layers are kept approximately perpendicular to each other. Here, the vertical traces are on one layer, and the horizontal traces are on another layer.

If you ask an engineer why you should use 45-degree turns, they'll probably give you one of two reasons. The first common explanation is that sharp angles will cause etching problems during manufacturing. This used to be true, but modern manufacturing techniques are good enough that this isn't really a concern anymore.

The second reason you'll hear is that right angles cause reflections or signal integrity problems, but there are only certain conditions under which a right-angled turn on a trace will hurt signal integrity. The issue is that a right-angled turn adds extra capacitance to the trace. How much extra capacitance it adds depends on the width of the trace and the PCB substrate. In many cases, the extra capacitance is so small that it isn't noticeable. You can estimate the amount of extra capacitance due to a single 90-degree turn using this rule of thumb: 1.7 times the width of your 50 Ω transmission line in mils is the additional capacitance in femtofarads (fF). For example, if your 50 Ω transmission line is 20 mils wide, then the extra capacitance from a 90-degree turn in that trace will be about 1.7 × 20 = 34 fF. Whether your design will care about an extra 34 fF depends on the rise time of your signals and their frequency. The

graph in Figure 6-6 shows the return loss of several example transmission line right-angle turns over frequency.

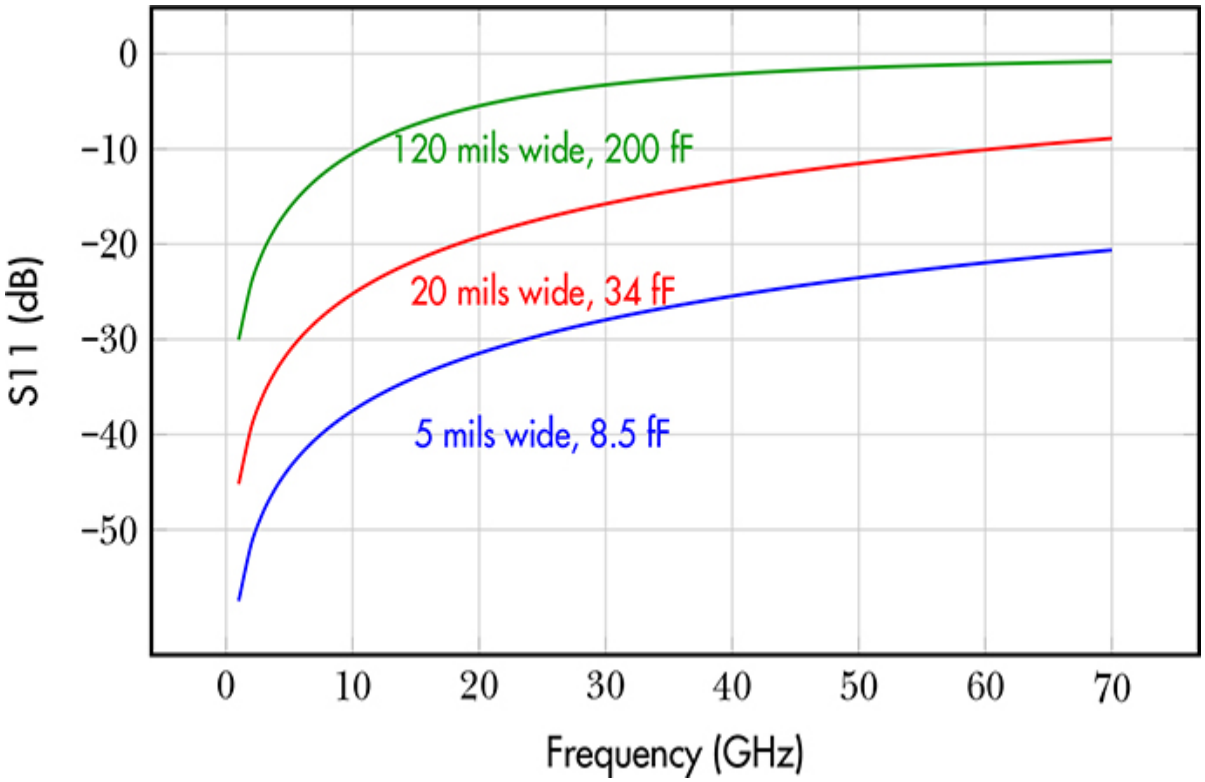


Figure 6-6: A simulation of the return losses of three different example transmission lines with a 90-degree bend

You can see that, for narrow transmission lines, performance doesn't start to suffer until the frequency is quite high. For wide transmission lines, this happens much sooner, but it's probably still at a high enough frequency that many designers won't need to worry about it.

Note that the 1.7x rule of thumb was derived assuming the substrate's relative permittivity is somewhere around 3.5 to 4, so it holds for substrates like FR-4 and RO4350B. If you use a substrate with a lower relative permittivity, the extra capacitance will be even less. If you use a substrate with a higher relative permittivity, the extra capacitance will be higher than the rule-of-thumb estimates. Another important factor to remember is that this is only the capacitance for a single 90-degree turn. If you meander a line back and forth with several 90-degree turns, all of that capacitance will add up, which can become a problem.

Even if your fundamental frequency is low, fast enough rise times can also be affected by 90-degree corners (since a signal with a fast rise time will contain high-frequency components). The rule of thumb here is if the width of your 50 Ω transmission line in mils is greater than five times the rise time in picoseconds, you shouldn't use 90-degree trace turns. As mentioned, you should also avoid unnecessarily fast rise times when possible for EMI reasons. We'll discuss this further in Chapter 8.

In short, most of the time, right-angle turns are fine. I happen to think 45-degree turns look better and make routing easier, so that's why I use them. Like in the case of schematic capture, take care to perform layout in an aesthetic and organized manner, keeping everything symmetric. Don't do this at the expense of performance, however. Only after you've met all of the performance requirements should you make changes that visually clean up the layout, and then check to make sure your changes don't subsequently affect performance.

Stackup Design

Just as critical as the components that go on a PCB are the characteristics of the PCB itself. The combination of the PCB materials, the physical dimensions of those materials, the traces, and other circuit features is called a *stackup*. Stackup design is one of the most important parts of designing electronics, but it's very rarely discussed in school, so lots of electrical engineers have a poor understanding of its nuances. This puts them at a real disadvantage, since having insight into stackup design can save time and money during fabrication, improve a design's performance, and make the layout process much easier.

Layers

The first place to start when specifying a stackup is the number of layers in the board. The more layers you use, the easier the routing will be, but the more expensive the board will be to fabricate. As such, you should use as few layers as you can reasonably get away with. Just be sure to pick an even number. You *can* fabricate an odd number of layers, but to do that, your fabricator is just going to make an even number of layers

and etch away all of the copper from one of the layers. This is much more complex than it sounds and will cause the board to actually cost more than if you added another layer to make the layer count an even number. A five-layer board will cost more than a six-layer board!

High-speed and RF designs often need at least four layers so you can dedicate a single layer as an uninterrupted ground plane, which is important for signal integrity reasons. Using a layer as a power plane can also be convenient, since routing power lines becomes as simple as dropping a via to the power plane. The large surface area of a power plane adjacent to a ground plane can also create a little bit of helpful capacitance and reduce ripple. RF designs often have a lower routing density than high-speed digital designs, so it's possible to get away with only a two-layer design with components and traces on layer 1 and a solid ground plane on layer 2.

Substrates

For many projects, stackup design doesn't extend much beyond picking the number of layers you want and then using the standard substrate and stackup your manufacturer offers for that number of layers. This will be the most cost-effective approach. Manufacturers usually offer a standard stackup for two-layer and four-layer (and sometimes six- and eight-layer) boards, typically using FR-4 as the substrate, and this will work just fine for most low-frequency to moderately high-frequency designs. You only start to get into more complex stackup designs if your PCB has high-speed or RF signals on it. The relative permittivity of FR-4 may be such that the dimensions of your transmission lines will be impractically large or impractically small. FR-4 is also quite lossy at high frequencies, necessitating other, more specialized substrates, which can be quite expensive. Sometimes manufacturers will offer standard stackups for high-frequency designs, too, but if not, you'll need to develop your own. That said, it's possible to design RF boards that operate in the 1 to 2 GHz range on FR-4 as long as you analyze and simulate your design to ensure you can meet your requirements.

For high-frequency or RF designs, the most important substrate properties to consider are the relative permittivity (written as Dk or ε_r,

and sometimes slightly erroneously called the *dielectric constant*) and the loss tangent (written as Df or δ). The *relative permittivity* is roughly a measure of how a given material stores an electric field. The permittivity of air is 1, and you can buy PCB substrates with a permittivity as low as 2 or as high as 10. The *loss tangent*, or *dissipation factor*, of a material is how much of an electric field is dissipated as heat. A lower loss tangent is better, especially at higher frequencies.

Brands like Rogers, Megtron, and Isola make a wide range of substrates that can have a very small loss tangent and almost any Dk value between 2 and 10 that you want. Take a look at these if you're trying to design an antenna, millimeter-wave, or high-speed circuit. If you're looking for a place to start, some popular RF substrates include 4350B and 4003C from Rogers. If you want specific guidance for what material to use, I highly recommend calling or emailing the substrate company you're interested in. Personally, I've had excellent support from Rogers. They offer free samples of their substrates, which are great for milling or etching on, and I was able to talk to an application engineer quickly. The engineer even offered to review my stackup and make suggestions. Similarly, your PCB fabricator can also review your stackup and make suggestions to improve manufacturability and cost.

Rogers also has a substrate in the same 4000 series family called 4835, which costs about the same as 4350B and 4003C but will oxidize slower. This only really matters for designs that must operate for many years, so you don't need to worry about this if you're just prototyping. Nevertheless, Rogers recommends 4835 for new designs.

While lots of different substrate options are out there, bear in mind that most PCB fabricators keep only a few regularly in stock. If your fabricator doesn't have the material you want in stock, you'll have to wait for it to ship, adding lead time to your schedule. To avoid this, try telling your fabricator beforehand that you'll be submitting an order using a particular material; they'll typically be happy to order it ahead of time.

If your transmission line calculations show that the conductor must be impractically wide, picking a substrate with a higher relative permittivity typically allows you to reduce the conductor width, which

may make the board easier to lay out. A higher relative permittivity can also be beneficial for antenna design, especially in reducing antenna size (but note that a higher Dk also causes higher insertion loss, since more of the fields are contained within the substrate). Conversely, picking a lower relative permittivity allows you to increase the conductor width, which can be helpful if your initial calculation shows that your trace needs to be extremely thin. The other factor you can try changing is the thickness of the dielectric above and below your impedance-controlled trace.

When designing impedance-controlled structures, make sure to use the relative permittivity in the datasheet that's marked as the design value, not the process value. Some PCB substrate companies, especially Rogers, will provide two different Dk numbers. This is because a material's apparent relative permittivity depends on how you measure it. In fact, IPC specifies over a dozen different measurement methods. Additionally, some substrates are anisotropic, and the location of the substrate weave can affect its measured relative permittivity. The *process value* is the number that the manufacturer measures on raw material to make sure batches of material are within the expected tolerance. The *design value* is measured by testing a stripline on the material, which will give a slightly different result than measuring the bulk material and will be closer to what your circuit will see. Use the design value or you may end up with a PCB trace that doesn't have the impedance you want.

There are two other factors that can affect apparent relative permittivity. First, trace surface roughness can change a signal's phase velocity at frequencies above about 20 GHz. Second, relative permittivity can also change with temperature, known as the TCDk (temperature coefficient of Dk). Unfortunately, the datasheet doesn't always include all of this information, but it's important to be aware of this effect. If your design is sensitive to small changes in relative permittivity, you may need to reach out to the manufacturer to get this data, search Google Scholar, or perhaps even measure it yourself.

Both the relative permittivity and loss tangent are frequency dependent. Usually they apply over a wide frequency range, so if the datasheet gives you only the relative permittivity at 1 GHz, it will

probably be about the same at 2 GHz. Don't use the value that's in big font on the first page of the datasheet. Find the curve showing Dk or δ over frequency and read the value from there.

When you're designing a custom stackup, layers can be made of two different types of materials: core and prepreg. *Core substrate* is a dielectric material with copper plating on both sides. *Prepreg* (short for *pre-impregnated*) is a dielectric material that's partly resin-cured and doesn't have any copper plating. After your design is etched, the layers are stacked together (hence, a *stackup*) and bonded. This process is called *lamination*.

There are two types of lamination: foil and cap. In *foil lamination*, the dielectric between layers 1 and 2 will always be prepreg. This is by the far the most common and also the cheapest method. Figure 6-7 shows an example four-layer foil lamination stackup.

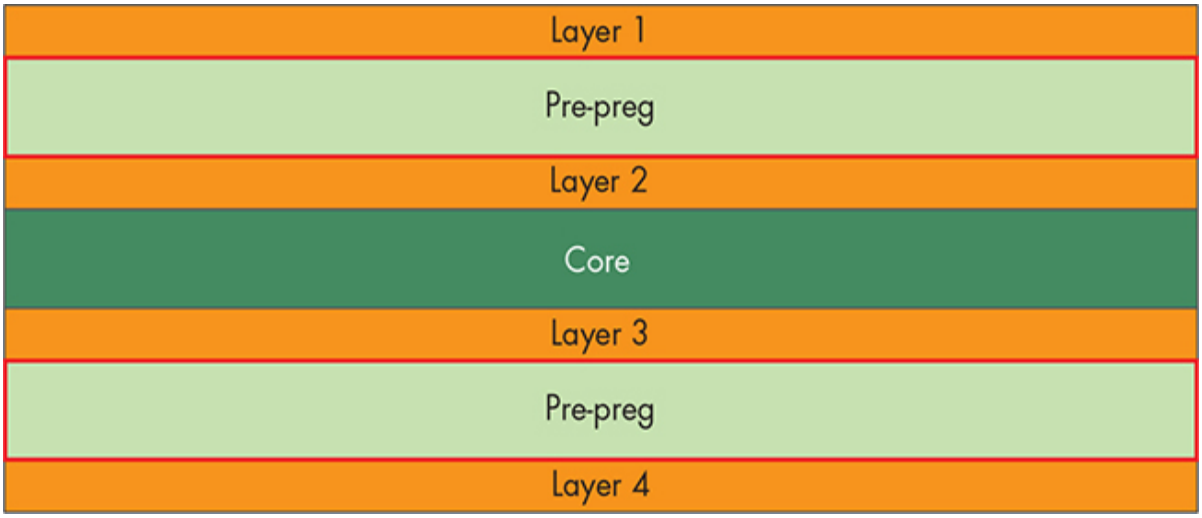


Figure 6-7: A very common four-layer stackup with prepreg on the outside. Note that the core consists of both the substrate in the middle and the copper foil of layers 2 and 3, whereas prepreg is just the substrate material and doesn't include the copper foil.

By contrast, *cap lamination* features core material between layers 1 and 2. This technique is used when you have blind vias between layers 1 and 2 and you're unable to use laser or controlled-depth drilling for some reason. It's also used when you need a special material between layers 1 and 2, like Rogers 4350. You can see an example of a stackup with cap lamination in Figure 6-8.

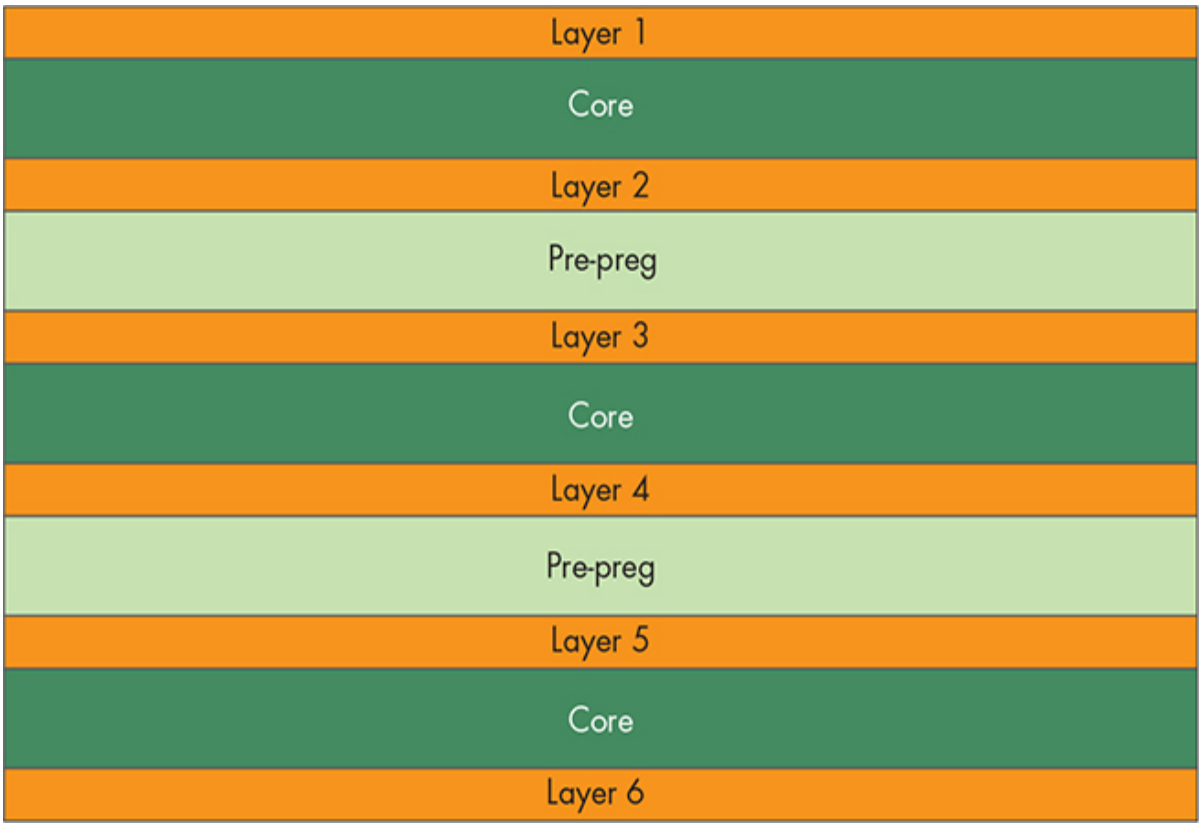


Figure 6-8: An example stackup of a six-layer board that alternates core and prepreg, starting with core on the outside

Alternating core and prepreg isn't the only way to construct a stackup. You can also put multiple sheets of prepreg on top of each other. Since core material comes copper plated and prepreg doesn't, you may think that you can route only on core layers, but it's completely possible to route on prepreg layers as well. To allow for this, layers of copper foil can be laid down between sheets of prepreg during construction. You should always talk to your fabricator to come up with a mutually approved stackup, including layer thicknesses and what your lamination will look like.

Whatever material you choose for your design will have both a core and a prepreg version. For example, if you choose FR-4, your PCB manufacturer will use FR-4 core (which contains fully cured resin) and bond it to other FR-4 core layers with FR-4 prepreg (which contains resin that's partially cured). As long as the materials are compatible with each other, you can stack different combinations of material to achieve your desired thickness. However, there aren't an infinite number of

thicknesses available, so you may need to combine several sheets of material to get to the thickness you want. Core materials in the Rogers 4000 family, for example, are available in 8-mil, 12-mil, 16-mil, 20-mil, 32-mil, and 60-mil thicknesses, as well as several different panel sizes. The prepreg in this family is available in 3-mil, 4-mil, and 5-mil thicknesses. There are a huge range of possible stackups and thicknesses you can achieve by mixing and matching those materials, and since they're all in the same family, you know that they'll all bond to each other. Just be sure to give your PCB a symmetrical cross section if you're combining materials of different thicknesses. Having an unbalanced or asymmetrical cross section can cause your board to warp during lamination.

Traces

If your design will be using a lot of current, you may want to change the copper thickness in your stackup. Curiously, copper thickness is expressed in ounces instead of mils. This refers to the weight of copper it would take to coat 1 square foot with the thickness of copper that you want to use. For example, a 1-square-foot layer of copper 1.4 mils thick would use exactly 1 ounce of copper. In other words, a PCB layer with a copper weight of 1 ounce means the copper is 1.4 mils thick. Table 6-3 shows some common weights and their equivalent thicknesses. In general, dividing the desired thickness in mils by 1.37 gives the weight in ounces.

Table 6-3: Common Weights of Copper and Their Thickness Equivalents

Weight (oz)	Thickness (mils)	Thickness (µm)
0.5	0.7	18
1	1.4	35
2	2.8	71
3	4.1	104

Weight (oz)	Thickness (mils)	Thickness (µm)
4	5.5	140

Most designs use 1-ounce copper on the outside layers and 0.5-ounce copper on inner layers. You can increase copper thickness as much as you want, but it will start to get very expensive. High-current designs may need to use something like 4-ounce copper.

Besides cost, there's another issue you'll need to contend with as copper weight increases: under-etching. This happens because the acid used to etch away unwanted copper doesn't attack all areas equally. Here are the steps to create an internal layer in a PCB stackup:

1. Choose a standard weight of copper that's available on the substrate you're using. This might be something like 0.5 ounce or 1 ounce.
2. A layer of etch resist is printed on the substrate to protect the copper you want to keep.
3. The substrate is then placed into an acid etching tank, which removes all copper not protected by the printed etch resist.
4. The substrate is then put into another tank to neutralize the acid and halt the etching process, remove the printed etch resist, and clean the surface.

This whole process will remove a small amount of the remaining copper. For example, if you chose a 1-ounce copper weight, which is about 1.4 mils, the internal layer printing and etching process will reduce that to about 1.2 to 1.3 mils. Figure 6-9 shows a cross section of an under-etched trace.

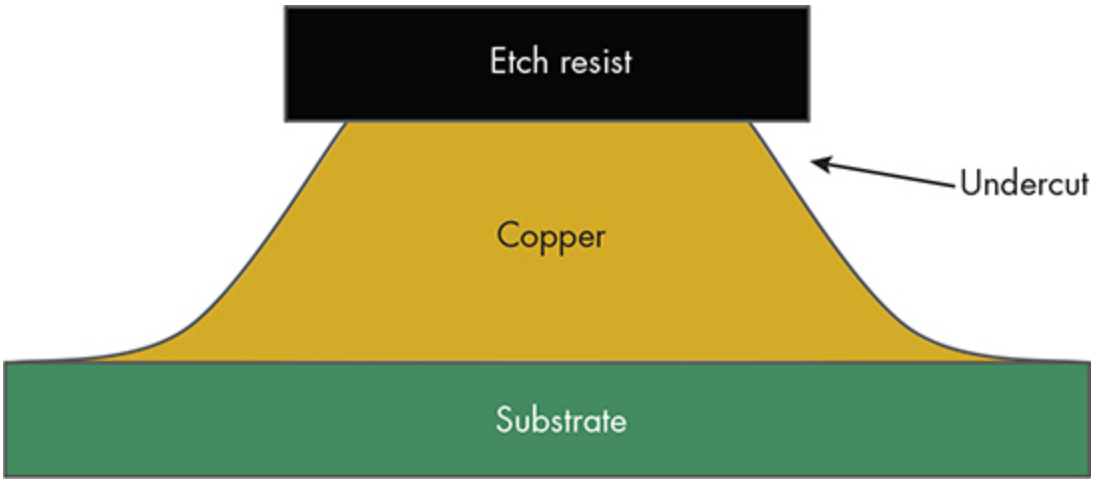


Figure 6-9: An example of an under-etched trace on an internal layer. A trace without this problem would have left and right edges that align with the left and right edges of the etch resist.

The process is different for the outside layers:

1. Once a lamination is complete, all vias are drilled.
2. Those holes now need to be plated. The fabricator chemically deposits a few microns of copper across the entire surface of the board, including inside of the vias.
3. A plating resist is printed on the board so that only the vias, traces, and pads are exposed.
4. The board is then put in an electroplating tank, which deposits copper only on the areas that are unprotected by the plating resist (vias, traces, and pads). This adds about 1 mil of copper to the walls of all drilled holes. Surface copper deposits faster than the copper in the holes, so you end up with about 1.3 mils of copper on the outside.
5. The plating resist is removed and an etch resist, usually tin, is added. This exposes the unwanted base copper and protects the vias, traces, and pads that we want to keep.
6. Finally, the board goes into an etching tank and the unwanted base copper is removed.

This final etching process also removes some of the copper that was electroplated, so the resulting cross section of copper on the board looks like Figure 6-10.

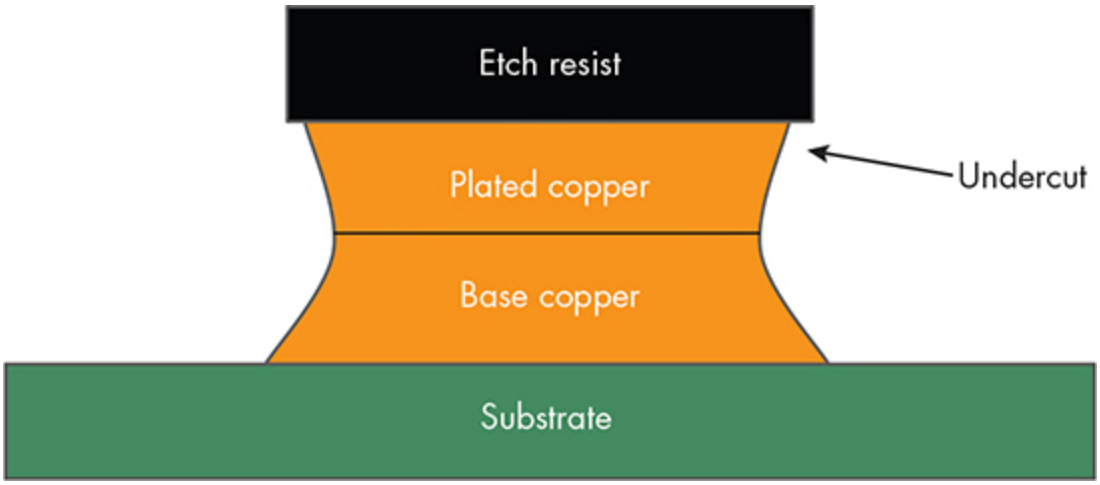


Figure 6-10: An example of an under-etched trace on an outer layer

In summary, when you're choosing the copper weight of your board for outer layers, remember that the finished thickness will be that weight plus an additional 1.3 to 1.4 mils of copper. This affects the minimum width of traces on the outer layer and also means that the impedance of a trace of the same width is different on an internal versus an external layer. If you need 4-mil or smaller traces on an outer layer, talk to your fabricator. They may require that you use a thinner copper weight.

Table 6-4 outlines the minimum trace and space widths for various thicknesses of copper. Your manufacturer should also advise you if they believe your copper thickness and trace width combination will lead to yield problems, where a significant number of boards in the production run are unusable. Use this table as a guide, but be sure to talk to your fabricator to get their approval for your desired minimum trace width and copper thickness.

Table 6-4: Copper Thickness vs. Minimum Trace and Space Width

Copper thickness (oz)	Minimum trace/space width (mils)

Copper thickness (oz)	Minimum trace/space width (mils)
6	25
5	20
4	16
3	10
2	8
1	6
0.5	3

When you send out your design, be sure to specify which traces need to be impedance controlled. For example, Figure 6-11 shows a stackup drawing sent to a fabricator. Notice how it calls out the intended impedance of traces on layers 1 and 2, along with their reference layers.

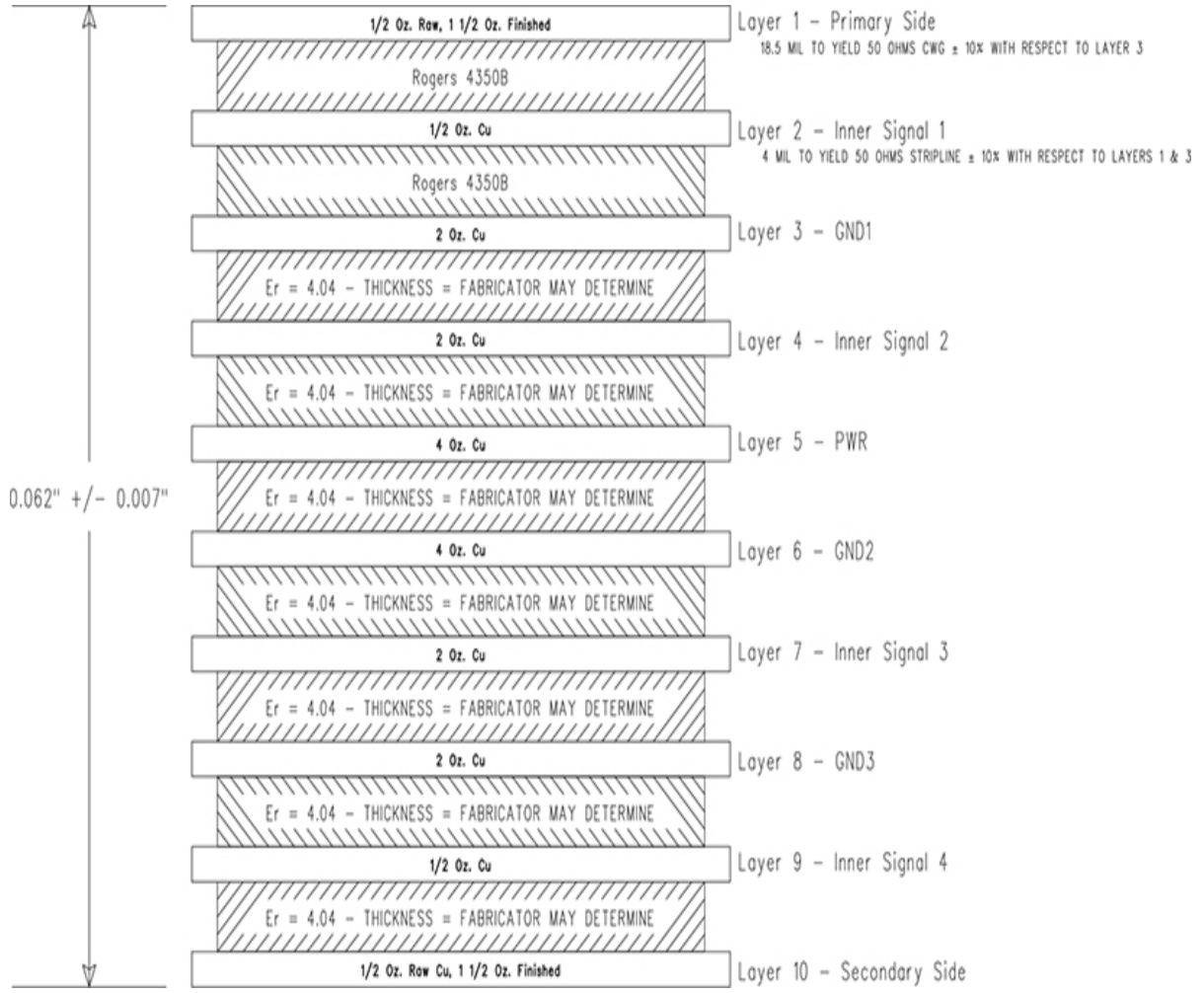


Figure 6-11: An example stackup of a 10-layer board with impedance-controlled traces on layers 1 and 2

Because of manufacturing variability, the fabricator will make small adjustments to the dimensions of your stackup to meet the impedance you specified, so don't be alarmed if you get your board back and find that your impedance-controlled trace is a couple mils thicker or thinner than you asked for. To make it easier for your fabricator to locate nets that must be impedance controlled, set those traces to all use a single unique width that isn't shared with any traces that aren't impedance controlled. Your fabricator should provide an impedance report with your PCBs that shows the new dimensions and the actual measured impedance.

Traces that are engineered to have a particular impedance are known as *transmission lines*. There are many different structures you can use to

achieve a particular impedance. Three popular options are *stripline*, *microstrip*, and *coplanar waveguide (CPW)*. A stripline transmission line has a conductor sandwiched between two ground planes, one above and one below. A micro-strip transmission line has a conductor on the top layer and a ground plane under it. A CPW is similar to microstrip, but it has a ground plane on each side of the conductor as well as underneath it. Figure 6-12 shows these different styles of transmission line.

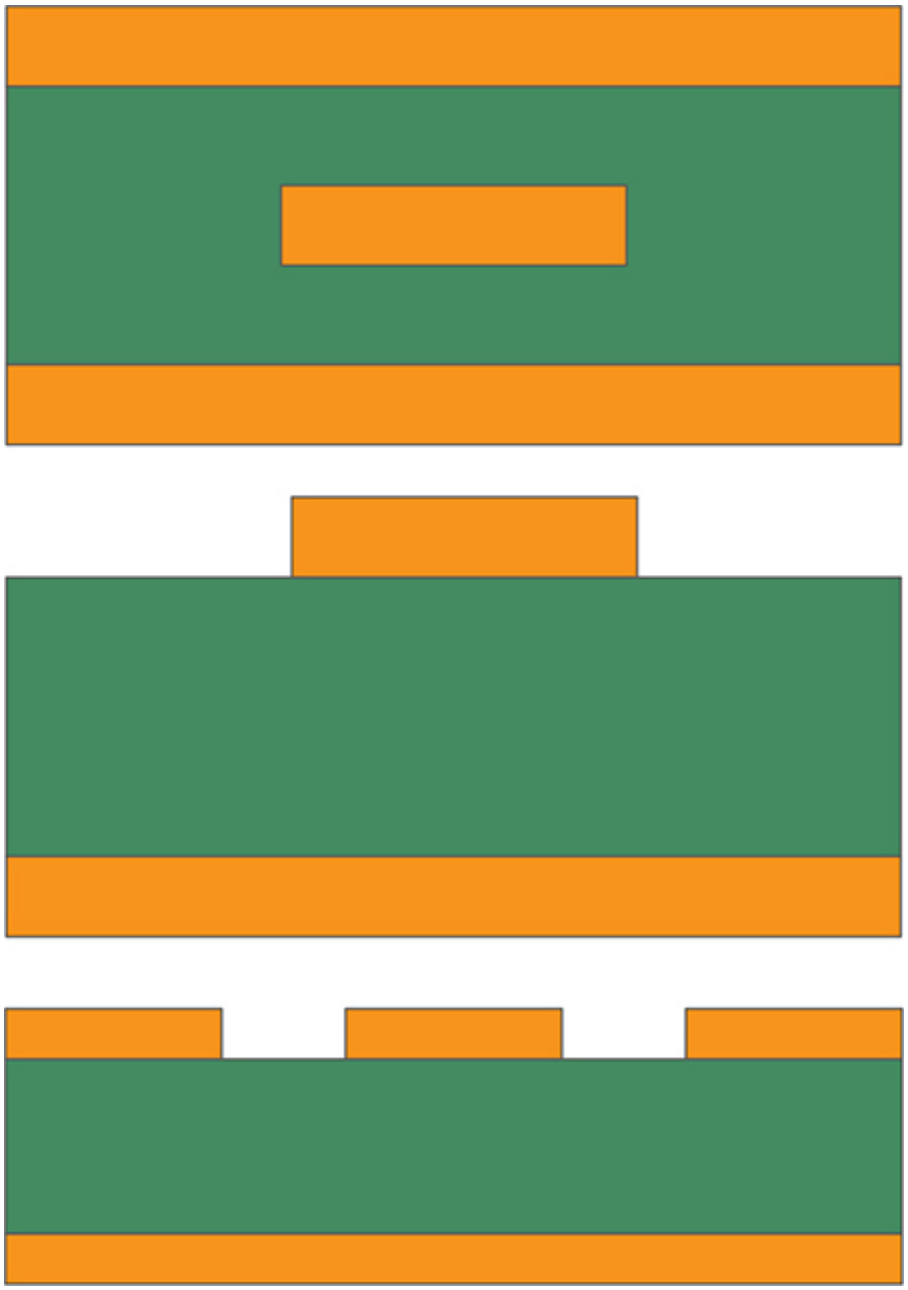


Figure 6-12: From top to bottom: Cross-sectional views of stripline, microstrip, and coplanar waveguide transmission lines

The impedance of each of these transmission lines depends on the exact dimensions and spacings of the copper and dielectric. Solder mask also affects the impedance of transmission lines, so you should either calculate the transmission line dimensions while taking the relative permittivity of the solder mask into account or remove solder mask from that area. There are plenty of online calculators for figuring out

the spacings you need to get to a particular impedance. You can find links to some good ones on this book's website.

One advantage of using microstrip or CPW transmission line structures instead of stripline is that the entire structure stays on the top layer, meaning you don't need to worry about via transitions (assuming all of your parts are on the same side of the PCB) and you'll have access to probe and modify the entire transmission line. Incorrect via transitions can degrade transmission line performance, and in the case of stripline, that's very hard to fix without respinning the board. Repairing microstrip or CPW transmission lines is also difficult without compromising performance, but it's much easier than having to excavate the PCB to even get access to the signal in the first place. You can also probe microstrip and CPW transmission lines using an E field or H field probe, which isn't possible with stripline because it's shielded above and below by a ground plane. On the other hand, because stripline transmission lines are well shielded and therefore better isolated, you can route them more densely, and they're less likely to pick up noise or crosstalk.

Given the same impedance, stripline is always going to be thinner than microstrip. But stripline also needs a very consistent relative permittivity across the entire substrate. This is because stripline is surrounded on all sides by the substrate material, and any change in the relative permittivity along the length of the trace will change the characteristic impedance of the transmission line and cause reflections and loss. Losses in microstrip tend to be less than losses in stripline because half of the fields in microstrip are in air rather than the PCB substrate, and air has a negligible dissipation factor compared to the substrate.

If you're prototyping RF designs using a circuit mill or etching your own PCBs, microstrip is great. You don't need to make any vias, and you only need to etch or mill one side of your copper-clad substrate. The precision spacing requirements that CPW has between the conductor and the top ground plane can be difficult to achieve with a mill or home etching, but microstrip doesn't have that problem.

Vias

Another important aspect your stackup should specify is the via structure. The simplest and cheapest designs contain only through-hole vias, which are just holes that are drilled and plated after the layers of the PCB are pressed and bonded together. Other, more exotic via structures are also possible: blind, buried, and microvias. Adding these kinds of vias to your board will add cost, but they also allow you significantly more design freedom. Some high-performance designs actually can't be implemented without them. Figure 6-13 illustrates the different kinds of vias.

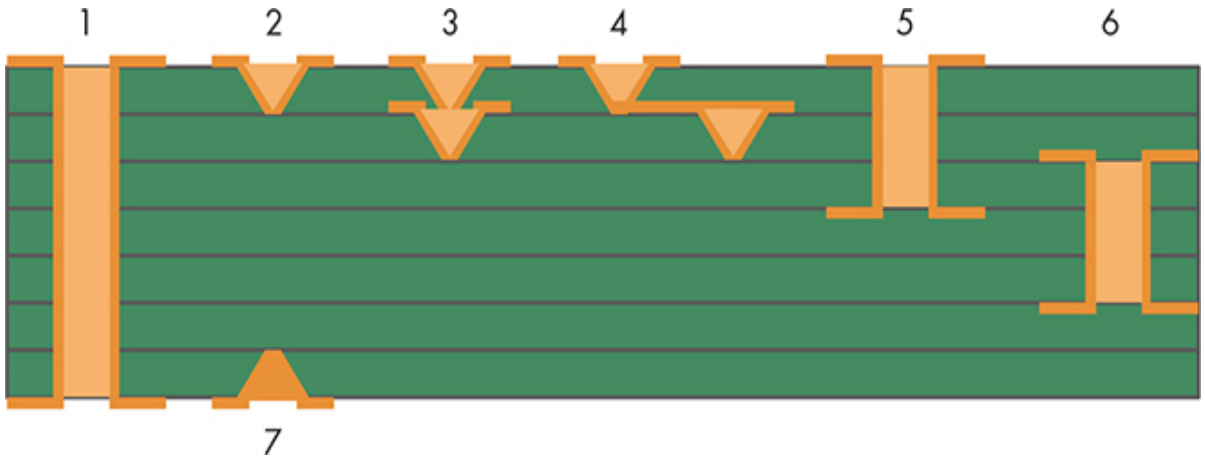


Figure 6-13: Different via types: a plated through-hole via (1), a microvia (2), a stacked microvia (3), a staggered microvia (4), a blind via (5), a buried via (6), and a copper-filled microvia (7)

A *through-hole via* (1 in the figure) goes all the way through the board. *Microvias* (2) are very small vias that usually connect only one layer to an adjacent layer. It's possible to stack microvias so they pass through additional layers (3), but this usually isn't a good idea: Stacking microvias directly on top of each other can cause delamination and other reliability problems in your PCB. Instead, you can stagger the microvias (4) such that the second microvia is offset from the first. This has been shown to improve reliability. *Blind vias* (5) start on the top or bottom layer and don't go all the way through to the other side. Meanwhile, *buried vias* (6) connect inner layers but don't extend to the top or bottom layers. Microvias can also be *copper-filled* (7), which allows

them to be placed directly on component pads for very dense routing. They also transfer heat more effectively than regular microvias.

Blind vias can substantially increase the cost of your fabrication and make it take much longer, depending on what layers they're on. Generally, it's best to keep your blind vias from overlapping on layers. This is because overlapping blind via layers takes higher precision since they have to be more carefully aligned. Depending on the design, it might also require another lamination cycle, which significantly increases fabrication time and cost. (A *lamination cycle* is the process of bonding all of the sheets of material together under high temperature and pressure to form a PCB. For more complex boards, the stackup is broken into subassemblies that are sequentially laminated together.) You should discuss your planned via sets with your manufacturer to make sure your design is easy to manufacture and that it minimizes lamination cycles. Sometimes a change that's inconsequential to you can make a big difference to your manufacturer.

If you want to save some time and money while still having something like blind vias, see if you can get away with *backdrilled vias*. These act like blind vias but are manufactured in a different way. They start as through-hole vias and are plated, but then the manufacturer uses a slightly wider drill to drill away half of the plating. The result is a hole that goes all the way through the board, but only half of it is conductive. You might want something like this if you're transitioning a stripline to a microstrip. If you use a through-hole via for an RF transition like that, you'll end up with a stub on your transmission line from the extra length of the via, which can cause matching problems. However, before you decide that a via stub will be a problem for you, do a little analysis to see if that's really true. In many cases, your design will work fine with a via stub. If you decide to use backdrilling out of fear rather than because you took the time to determine you actually need it, you'll end up paying a lot more for your boards than you need to.

Drilled buried vias, those that go between two internal layers, always add a lot more cost to your PCB fabrication since they always require multiple lamination cycles. Drilled blind vias also add a lot more cost, but microvias tend to add a little less cost.

Placing a single via between two unique layers adds a drill step to the manufacturing process. This means that it's the same price to add a single via between two layers as it is to add 1,000 vias between those same layers. As such, if you're asking for expensive blind vias between layers 1 and 2, only asking for a couple of them isn't going to save you any money. You'll have already moved up to a more expensive pricing tier and longer lead time, so either try to eliminate the vias or take advantage of the extra step and add as many vias as you want.

The size of the holes in your vias is one of the main drivers of the cost to manufacture your PCB. Most manufacturers limit the hole and via size in their standard stackups to 6 mils or greater in diameter. Smaller holes require a laser to drill since mechanical drill bits that small can't reliably plunge without breaking. Laser drilling is typically only used for microvias. Be aware that adding laser-drilled vias to your design will increase the cost.

Vias have a maximum depth-to-width aspect ratio. Regular drilled vias can have an aspect ratio of up to 10:1, meaning the hole can be drilled up to 10 times deeper than the via is wide. For example, a hole 10 mils in diameter can be drilled through up to 100 mils of substrate. Some fabricators can even go up to 20:1. For laser-drilled vias, the aspect ratio is much smaller, usually 2:1 or 1:1, depending on the fabricator. That means that if you want to laser drill a 6-mil via, you must do it on a substrate no thicker than 6 mils (for an aspect ratio of 1:1). Keep in mind that the maximum substrate thickness here refers to the substrate that the via must traverse, not necessarily the entire board thickness. You might have a 4-mil buried laser-drilled via that's going from layer 4 to layer 5 through a 4-mil prepreg, which is fine even if your finished PCB thickness is 62 mils.

In addition to the depth-to-width aspect ratio, vias must also have a minimum annular ring size. This size is specified as the distance between the edge of the intended drill hole and the edge of the copper pad that's going to be drilled into. Drills don't have perfect accuracy, and if they don't hit exactly in the middle of the via pad, they could "break out" and land partly outside the intended via location. Making the annular ring larger allows for a less accurate drill hit to still be

functional and can be improved even further by adding a teardrop tapering along the side, as shown in Figure 6-14.

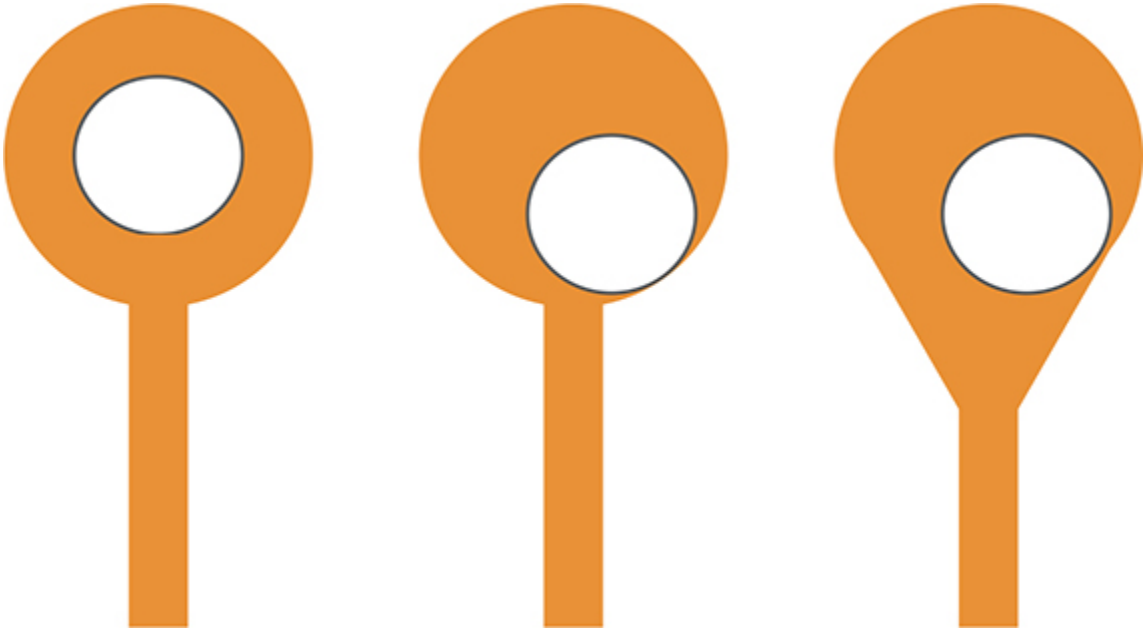


Figure 6-14: A center drill hit on the annular ring (left), a breakout (center), and a taper preventing a breakout (right)

A good rule of thumb to size your pad to prevent breakout is to make the pad twice the diameter of the finished, plated hole. Some fabricators may ask you to make the pad even bigger than that, but most are okay with a 2:1 ratio.

The minimum annular ring size for a via will depend on the copper weight of the layers it's on. For typical copper weights (between 0.5 and 1 ounces), a good rule of thumb is to use an annular ring of at least 6 mils. Talk to your fabricator to be sure.

Vias can cause assembly problems if you're not careful. If a via is on a component pad, solder can wick down into the hole during reflow, not leaving enough for the component to be firmly attached to the pad. There are two ways to avoid this. The cheapest way is to *tent* your vias. This is a process where you apply solder mask over the tops of vias on both sides of the board (but not on component pads). Wet solder mask has enough surface tension to cover the hole and prevent anything from getting into it. This way, the via in a component pad on one side has a thin membrane of solder mask on the other side, creating a well. The

risk with tented vias is that there can be chemical entrapment inside the via, which over time can cause failure. Tenting vias is fine for prototyping purposes, but you may want to consider it more carefully before building a production design with them.

The safer solution is to *plug* your vias by filling the holes with a non-conductive or conductive epoxy. They can also be filled with metal. When plugged vias are used on a pad, they're sometimes called *active pads*, and the vias are plated over after being plugged. Other terms you'll hear for this are *via-in-pad* or *via-in-pad plated over*. Plugged vias ensure nothing can wick down into the via and that nothing can get trapped inside it. Of course, this technique adds time and cost to your fabrication.

The best way to avoid this extra time and cost is not to use via-in-pad, unless they're microvias spanning only to layer 2 or layer 3. It's possible to design even very dense, complex boards without using via-in-pad. The one exception to this is very fine-pitch BGA parts that require it. Avoiding via-in-pad will not only make assembly easier but also make rework and repair significantly easier. That will make your technicians happier, which is always a good thing.

A Stackup Example

Let's put what we've discussed about stackup design into context with a real-world example. Figure 6-15 shows a fabricator's report about an actual, slightly complex stackup. This board was for a transmitter device that contained power, RF, and digital elements.

Layer	Type	CU Weight	CU %	Material Description	Via Structure	Segment	Glass Style	Material Family	Copper Plating Thickness [mil]	Thickness after lamination [mil]
Soldermask										0.80
L1	Mixed	H	100							2.00
				4.0 mil H/H		Core		RO4350B		4.00
L2	Signal	H	100							0.60
				Press thk = 7.50 mil		Prepreg	4450F(1)	RO4350B		7.50
							4450F(1)	RO4350B		
L3	Plane	2.0	100							2.40
				3.0 mil 2/2		Core		IT180A		3.00
L4	Signal	2.0	20							2.40
				Press thk = 7.74 mil		Prepreg	1080HRC(71)	IT180A		7.74
							1080HRC(71)	IT180A		
							1080HRC(71)	IT180A		
L5	Plane	4.0	70							4.80
				4.0 mil 4/4		Core		IT180A		4.00
L6	Plane	4.0	70							4.80
				Press thk = 7.74 mil		Prepreg	1080HRC(71)	IT180A		7.74
							1080HRC(71)	IT180A		
							1080HRC(71)	IT180A		
L7	Signal	2.0	20							2.40
				3.0 mil 2/2		Core		IT180A		3.00
L8	Plane	2.0	70							2.40
				Press thk = 4.20 mil		Prepreg	1080(65)	IT180A		4.20
							1080(65)	IT180A		
L9	Signal	H	20							0.60
				4.0 mil H/H		Core		IT180A		4.00
L10	Mixed	H	40							2.00
Soldermask										0.80

* Estimated Cu Plating for reference use only.

Specification (Over mask on plated copper:):	mil
Overall Board Thickness:	66.00
Tolerance:	+6.6/-6.6
Min-Max Board Thickness:	59.4-72.6

Anticipated Board Thickness:	mil
After lamination:	66.78
Over mask on plated copper::	71.18

Impedance Table

Layer	Impedance Requiremnt [ohms]	Tolerance [ohms]		Type	Upper Ref	Lower Ref	Designed Line Width [mil]	Plotted Line Width [mil]	Designed Spacing [mil]	Coplanar Spacing [mil]	Finished Line Width [mil]	Finished Spacing [mil]	Impedance Simulation [ohms]
		+	-										
L1	50	5.0	5.0	Coated SE CoPlaner	--	L3	18.50	19.25	--	20.00	18.50	--	51.6
L2	50	5.0	5.0	Single-Ended	L3	L1	4.00	5.25	--	--	4.75	--	49.9

Remarks:

Please Note: The stackup may change if the final manufacturing data is different from the information used to create this stackup

Mat Typ	Material Description	Rsn%	PNL	1 Pnl	Notes
Core	IT180A - 3.0 mil 2/2		16x18	2	
Core	IT180A - 4.0 mil H/H		16x18	1	
Core	IT180A - 4.0 mil 4/4		16x18	1	
Prepreg	IT180A - 1080	65%	16x18	2	
Prepreg	IT180A - 1080HRC	71%	16x18	6	
Core	RO4350B - 4.0 mil H/H		16x18	1	
Prepreg	RO4350B - 4450F	1%	16x18	2	

Drill Progs	Technology	Depth
Drill1	Mechanical	66.78
DRILL 1-3	Laser	12.70
DRILL 1-2	Laser	4.60

Figure 6-15: An example stackup of a 10-layer board with impedance-controlled traces on two layers

The report shows a 10-layer board using two different substrates. Layers 1 and 2 use the Rogers 4350B. This cheap, general-purpose

substrate is good for RF applications because of its low loss tangent. The rest of the board is a substrate called 180A, which is basically the same thing as FR-4. The reason for these two materials is that layers 1 and 2 both have impedance-controlled transmission lines on them. They're both specified as 50 Ω, and one is a coplanar waveguide while the other is stripline. The ground plane references that they're using are noted, and the dimensions required to hit the 50 Ω goal are also listed.

There are three kinds of vias on this board: through-hole, a blind via from layer 1 to layer 2, and a blind via from layer 1 to layer 3. Notice that the board is 66 mils thick, which is thicker than the standard 62-mil board. This is because while working with the PCB fabricator, it was determined that a 62-mil board would result in one of the lamination layers being too thin. This same discussion also revealed that the aspect ratio of some of the blind vias was too small and the hole size needed to be expanded slightly.

The PCB was also fabricated with a different, second PCB manufacturer who gave different design notes. They suggested expanding some of the vias so that they could be mechanically drilled instead of laser drilled, which saved significant cost on the fabrication. The message is clear: Talk to your manufacturer early and often, and make sure it's a two-way conversation. They're experts at building boards. They know what will work and what won't, and they'll save you significant time and money if you let them. It's also a good idea to talk to more than one fabricator, since there's almost always more than one way to make your design. This is a great learning opportunity and will usually save you time and money.

Thermal Considerations

Nothing is perfectly efficient, which means your designs are going to generate heat. Since this heat is usually generated inside IC packages, the first step in thermal management for electronics is usually to try to efficiently conduct the heat out of that package and into something that can either move the heat to another place or dissipate it. To that end, chips that get hot almost always have a thermal pad underneath the chip

or a large exposed thermal tab that you can couple to your thermal management system. It's common to design vias that touch one of these thermal pads to conduct heat from one layer of a PCB to another, like those shown in Figure 6-16. These are known as *thermal vias*.

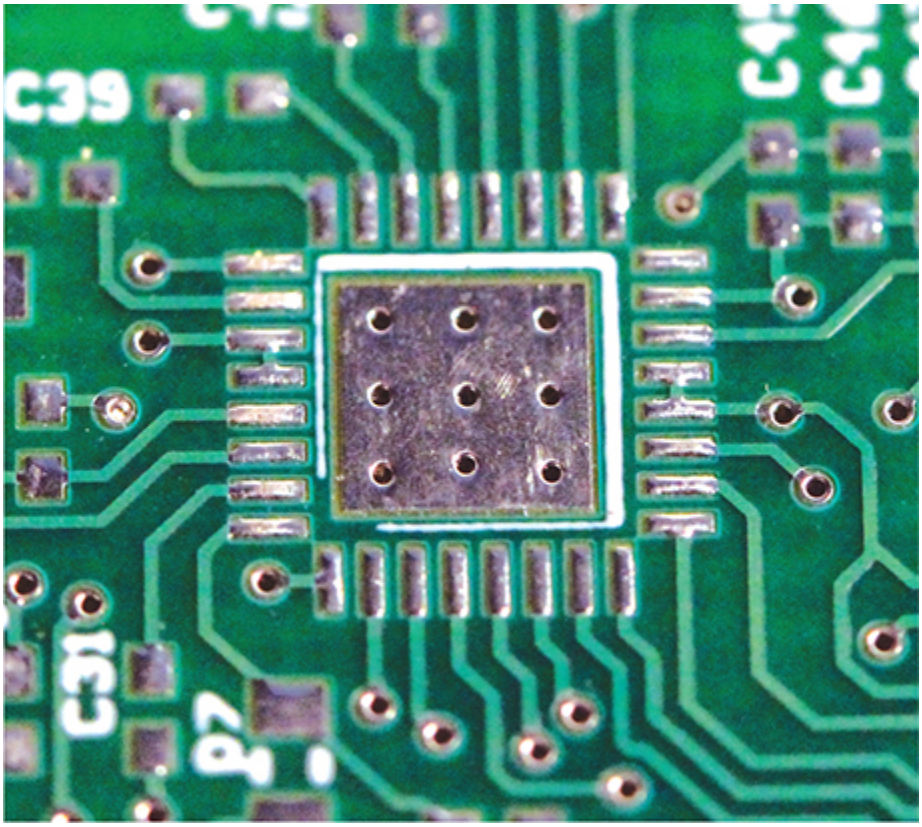


Figure 6-16: Vias can be seen here on the center pads of a component footprint.

Thermal vias help draw heat away from the chip and spread it out over the rest of the PCB. Traditional wisdom has held that you need a lot of them all right next to each other, and component datasheets usually show a recommended number and placement of thermal vias. However, more recent research has revealed that what is actually most important is making the chip's thermal pad large and having a large ground plane on the layer directly below it. Thermal vias themselves quickly have a diminishing return in improving heat transfer, since a lot of the heat is actually conducted through the dielectric substrate itself. You still need some vias on the thermal pad but not nearly as many as datasheets may suggest. Reducing the number of thermal vias will make assembly easier.

NOTE

For more information on the correct use of thermal vias, see PCB Design Guide to Via and Trace Currents and Temperatures *by Douglas Brooks and Johannes Adam (Artech House, 2021).*

You'll also see recommendations for thermal vias on many RF parts, as a very low inductance path to ground is critical for correct operation. Chips that call out the use of thermal vias are designed to use them; don't assume you can put a heat sink on top of the IC package instead. You can fill thermal vias with solid copper, forming a plug, which can help thermal transfer and make assembly easier. However, you will actually get better thermal performance from a larger diameter via with higher copper weight, even if it's not plugged. According to a presentation by Summit Interconnect, a normal 10-mil via with 2-mil thick copper plating has a thermal resistance about 12.5 percent lower than an 8-mil via filled with solid copper. Doubling copper thickness approximately doubles thermal conductance.

PCB manufacturers can also insert copper "coins" into your board, which are solid copper slugs to help move or spread heat between layers. These make your stackup more expensive but are more effective than moving heat through thermal vias or dielectric material.

Thermal vias can be used with normal PCB substrates like FR-4, but in extreme cases you may need to use a metal-core PCB (MCPCB). This is an expensive technique and should be used only if other thermal management methods fail and you have no other options. MCPCBs are sometimes seen on designs with high-power LEDs. If your design incorporates high-speed or RF parts, a stackup with a metal core may not be possible.

Another tool for engineering heat transfer is the *thermal relief*. Thermal reliefs also make it easier to populate and remove components when hand-soldering (with either an iron or hot air). Figure 6-17 shows an example of what thermal reliefs on vias look like.

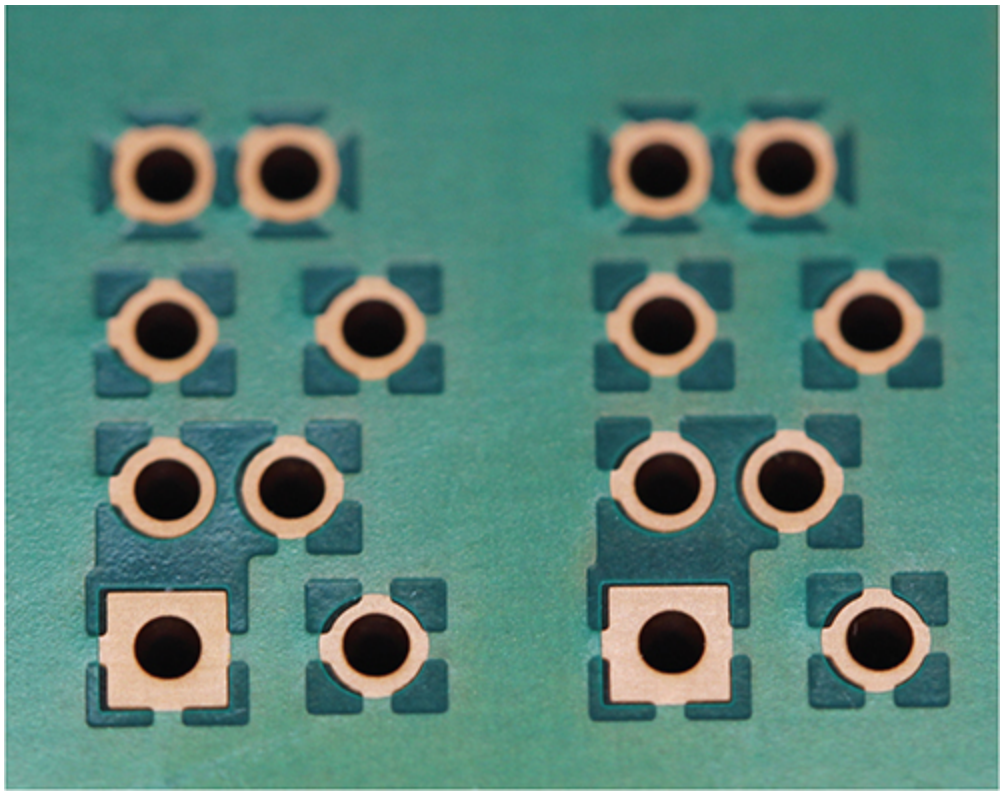


Figure 6-17: Vias with thermal reliefs

Ensure that there are at least three struts to the main polygon. Using thermal isolation where possible makes soldering and desoldering much easier, since it reduces the effective thermal mass of the component.

Use thermal reliefs on parts that are connected to large copper planes or polygons. This is especially important for preventing tombstoning during reflow on discrete components like resistors, capacitors, and inductors. Figure 6-18 shows an example of a tombstoned capacitor.

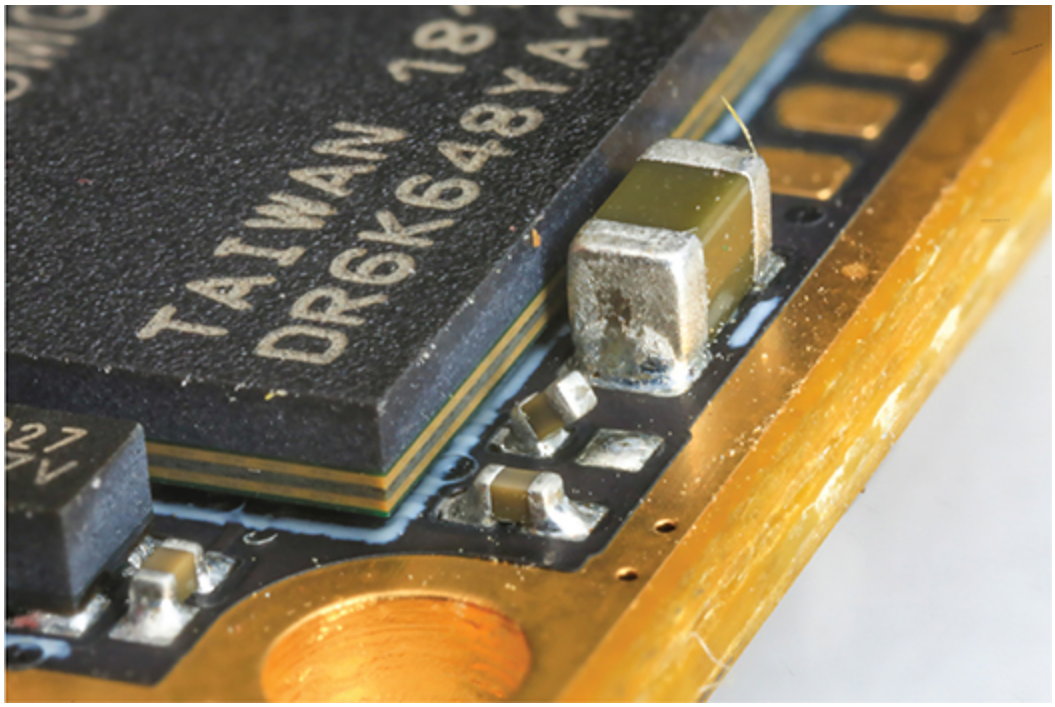


Figure 6-18: This capacitor has tombstoned due to lack of thermal reliefs. Photo courtesy of Greg Davill.

Notice how the middle capacitor in the figure is only soldered down on one end. The tombstoning occurred because the thinner copper trace heated up faster than the copper ground plane, and the surface tension of the solder was enough to pull the entire component upright. This is likely to happen any time an SMT part has a thick trace on one pad and a thin trace on the other. Thermal reliefs prevent this.

Pay attention to the placement of parts that have a large thermal mass. During soldering, if a component absorbs a lot of heat and it's right next to much smaller components, the smaller components may take longer to heat up as the heat is drawn into the larger component. This is especially important to consider if you're assembling by hand without a reflow oven, since it will be very difficult to heat up a large area all at once. Thermal reliefs can be useful here, too.

You have a lot of options to move heat around and dissipate it. The two most obvious solutions to remove heat are heat sinks and fans. They can be used together (a *fan sink*) for an even higher thermal-handling ability.

A heat sink or fan is effective only if it's thermally coupled to the source of the heat, however. This is typically accomplished with a thermal interface material, which can reduce the thermal resistance between two objects. One common thermal interface material is *thermal paste*. This is usually a white, greasy paste that's evenly distributed in a thin layer between a heat source and a heat sink. There's also something called *thermal epoxy*, which is a permanent, two-part adhesive that has a high thermal conductivity. Thermal paste is easy to remove if you ever need to replace or repair parts, while thermal epoxy is much harder to remove and will leave hardened residue stuck to both surfaces.

Graphite pads are another thermal interface material. These are thin sheets of thermally conductive material, usually with one or both sides coated in a weak adhesive, that can be cut to size with scissors. Graphite pads are available in several different thicknesses. One thing to watch out for with graphite pads is that they can be weakly electrically conductive. If you use graphite pads normally, this probably won't affect you, but I once tried to slide some thermistors down between a heat sink and a PCB that had a graphite sheet between them and ran into problems when small pieces of the graphite sheet caused my thermistor values to be wrong.

A good compromise between thermal paste and graphite pads are *gap pads*. These are soft sheets that are impregnated with thermally conductive material and are designed to eliminate any insulating air gaps between the heat-producing object and the heat sink.

You may need via plugging or via filling to prevent thermal interface material from getting into via holes. If you aren't using thermal interface material, you may need a dielectric barrier to prevent shorting between your PCB and metal heat sink. For designs that use high voltage, there may be minimum dielectric spacing requirements between your traces and the heat sink to prevent arcing. Finally, any PCB that will be attached to a heat sink should be constructed using a symmetric, balanced stackup to minimize board twist and warpage so that the PCB and the heat sink make complete contact.

Sometimes you need to move heat before you can dump it out to a heat sink. The best way to do this is with a *heat pipe*, a thin, flat copper

tube that's sealed at both ends. Inside is usually a little water. Heat pipes are great at moving heat from a small surface (like the top of an IC) into a larger heat sink. They can be fairly low-cost and are great for applications where there isn't enough room for a heat sink directly on top of the heat source.

It's very important that you make sure there isn't any air or empty space between a heat source and the rest of the thermal system. Air is a great insulator, which means you'll have a large thermal resistance in your thermal system and all of your carefully designed heat dissipation methods won't be doing any good.

If your design has parts on both the top and the bottom of the board, check for instances where large solder pads on the top layer sit directly above large solder pads on the bottom layer. This can cause heat transfer problems during reflow. Similar thermal problems can occur during the product's lifetime if both chips are trying to sink heat into each other. To avoid this, offset the footprints sightly. A large device with a high thermal capacity can also thermally "shadow" a smaller device next to it, which can prevent the smaller part from getting hot enough to properly reflow.

One more technique for dissipating heat is to use a thermally conductive resin or potting material. If you're planning on potting your design (see Chapter 12 for more information about potting), choosing a thermally conductive material can let you treat your entire enclosure and all of the potting material as a heat sink. Heat will be absorbed into the potting material and efficiently conducted between the heat source and the enclosure that the potting material is making contact with.

Overall, to determine the necessary thermal management materials and techniques for your design, you'll need to perform a full analysis of your thermals. A great resource that walks you through how to do this is the TI Application Note AN-2020, *Thermal Design by Insight, Not Hindsight* (*https://www.ti.com/lit/an/snva419c/snva419c.pdf*). Rather than reproducing the contents of that application note here, I recommend you read it and use it as a resource to help you calculate each piece of your thermal circuit and determine what you need and what you don't.

Common Gotchas

There are many factors to consider and remember when designing a layout, but there are also specific sources of mistakes that seem to trip up everyone. These gotchas often have nothing to do with physics or manufacturing; instead, they are often the result of stupid software design or non-obvious conventions.

An all-too-easy source of design mistakes is your EDA package's auto-router. This tool may seem like a great way to save time by letting the computer do the often tedious job of layout for you, but don't be fooled: Autorouters universally suck. They'll make poor decisions about where to route traces, even if your placement is good. The autorouter just doesn't know about all of the design considerations and requirements that you, the designer, are aware of.

Mistakes are also likely to creep in when importing or exporting your design files between the many CAD formats. Be sure to check the result carefully each step of the way. When an import doesn't work correctly, it can lead to subtle problems like missing solder mask, keep-outs, polygons, and silkscreen. Check everything.

Likewise, after you've exported the Gerber files, open them in a Gerber viewer like gerbv or ViewMate. (Both of these tools can be found on the book's website.) Some CAD software may not automatically repour polygons and flood fills when you expect it to, and polygon pour errors can cause shorts. Check for shorts or leftover thermal isolation, especially on inner layers, since those are the hardest to rework. Additionally, someone else who didn't draw the layout should review the completed layout and Gerbers.

Another way to avoid mistakes is to print out your design on a piece of paper at a 1:1 scale before sending out the PCB for fabrication. If you have the housing for the PCB already done or mocked up, you can do a fit check and make sure the holes all line up. If you have any of the parts in your BOM on hand, you can line them up with the printed-out footprints and make sure they're correct. You can also use this paper model to check connector orientation and pinouts to make sure everything will mate correctly. If you want to be even more precise, you

can 3D print your PCB before fabrication. Many CAD tools will allow you to export a 3D model of your PCB and include 3D models of the components on it.

One potential trip-up for new designers concerns the units commonly used in PCB layout. When someone refers to a *mil*, they mean 0.001 inches (one-thousandth of an inch). If you've ever worked in a machine shop, you may have heard this unit referred to as a *thou*, short for *thousandth*. The mistake that some people make is assuming that *mil* is short for *millimeter*. In the world of electronics design, it's therefore important to never shorten the word *millimeter* to *mil*, or you will really confuse people. (If you need an abbreviation for *millimeters*, use *mm*.) The mechanical drawings in datasheets for the component are almost always in mils, millimeters, or inches. A good drawing should tell you what the units are, so pay attention.

Speaking of units, make sure to use a consistent grid unit during layout. If you switch back and forth between imperial and metric, or if you change the grid size often, it's easy to end up with components that are offset from the grid, which can cause annoying alignment problems during routing.

DRC Checks

You should always use a *design rule check (DRC)* to ensure your design meets all the necessary manufacturing constraints. When you run your DRC, your board should pass with no errors. If there are errors or warnings that you don't fix, you should have a good reason for every instance, and it should be documented.

If the PCB fabricator doesn't provide a DRC file, you'll need to create your own using the design rules listed on the company's website. Using the existing or suggested rules will help keep costs down. That said, once you're producing boards in the thousands, or even in the hundreds, custom design rules won't have as big of an impact on your board cost.

Make sure your DRC is set up to run all checks in batch mode. DRC rules can be checked continuously as you route traces (known as

online mode) or only when you click the DRC Check button (known as *batch mode*). In some CAD software (like Altium), it's possible to accidentally check only a subset of the DRC rules, even after running both batch and online DRC. In Altium, you can see all DRC rules under Tools ▸ Design Rule Check ▸ Rules to Check. You might be surprised to find that some rules aren't assigned to be checked in either mode.

Keep in mind that you can use other DRC checkers besides the one that comes with your CAD program. For example, Mentor Graphics offers a program specifically for importing your design and running design checks on it. It can perform more in-depth checks and even do simulations on your design to find mistakes that your CAD program's DRC might miss. There are links to several DRC checkers on this book's website.

Footprint Mistakes

Drawing your own footprints for components can be time consuming and error prone. Some component manufacturers provide footprints for you to download that you can trust to be correct. One place to find these is a website called UltraLibrarian. You can search for and download footprints and models for a huge number of parts in a number of different formats.

Another place to find footprints that are already drawn is a website called SnapEDA (recently rebranded as SnapMagic). This site provides schematic symbols, footprints, and 3D models for millions of parts, and it will let you download those files in the native format of whatever CAD program you're using. If they don't have the footprint you need, you can pay them a small fee to draw it for you. You can find a link to SnapEDA and other resources on this book's website.

Whenever you download a footprint that you didn't draw, no matter where it comes from, you absolutely need to check that it's correct. Verify every detail—the dimensions, the silkscreen, the aperture, the pin assignments, the schematic symbol, the paste mask—against the recommendations of the datasheet. If a datasheet doesn't specify a particular parameter (like the paste mask, for example), use the value taken from the relevant JEDEC standard.

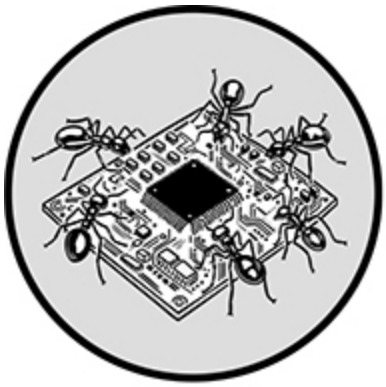
Keep in mind that while a datasheet may have only a single recommended footprint, there can be multiple correct footprints for the same part. Some manufacturers will offer several different versions of the same footprint: small, medium, and large (sometimes called high density, medium density, and low density). These differ based on how much extra space each pad has around the pin itself. A smaller version of the footprint is good if space is at a premium and you can get away with using only a small amount of solder paste. A larger version of the footprint is good if you're going to be assembling the board by hand and need to get a soldering iron tip on the pads.

It's exceptionally easy to make a pin assignment mistake on small three-pin MOSFETs and BJTs, so give extra scrutiny to these components. There are many variants of the same package with different pin mappings and very subtle naming differences in the part that are easy to miss—often only a single letter in a 14-character manufacturer part number. One way to avoid mistakes on those parts is to name the pins and pads B, E, and C (base, emitter, and collector) for a BJT, or G, S, and D (gate, source, and drain) for MOSFETs. Using this naming convention instead of numbers will let you reuse the same schematic symbol for multiple footprints while avoiding pin-to-pad mapping errors.

Conclusion

A successful layout will result in a product that isn't overly expensive, performs as expected, passes regulatory requirements, and lasts a long time. Conversely, messing up your layout not only will cause your device not to work as expected, but also the resulting problems can be difficult to debug. Use this chapter like a checklist to make sure you've remembered everything as you're looking over your layout. Design reviews are really helpful for heading off layout problems, too; the more people who scrutinize your layout, the less likely it is that something will slip by.

7
DESIGN FOR EXCELLENCE



People use different *DF* acronyms to refer to design considerations for particular areas, like design for manufacturing (DFM), design for test (DFT), design for inspection (DFI), design for variability (DFV), design for reliability (DFR), design for cost (DFC), and many more that may or may not apply to your situation. Collectively these are known as *DFx* (*design for X*, or sometimes *design for excellence*). This chapter addresses particular issues you should watch out for that relate to DFx.

Design for Fabrication

More than once, I've seen engineers get all the way through the design process, submit for fabrication, and then have to redesign large portions of the board because they didn't consider what was actually manufacturable. Just because something looks great in simulation doesn't mean it's realistic to build. Many schedule slips happen because when we plan, we assume the time between "design is done" and "design has started fabrication" to be only a couple of days, when in fact issues that your fabricator raise can take weeks to resolve. Designing for fabrication will reduce frustration for both you and your fabricator. In

this section, we'll talk about how to make sure your design can be built reliably, cheaply, and at high volume.

Fabricator Constraints

Make sure you understand the manufacturing abilities of your PCB fabricator and design with those constraints in mind. Manufacturers are able to fabricate extremely small trace widths, complex via structures, and more, but these sorts of exotic features will be very expensive. If you can design within the standard specifications your PCB manufacturer offers, you can get your boards back at a lower cost and in less time. For example, say you design a trace to be 9 mils wide, but your manufacturer has a standard or low-cost tier that requires a minimum 10-mil trace size. If you really need 9 mils, that's one thing, but if 10 mils would also be acceptable, just go with 10 mils. Otherwise, your design will be more expensive than necessary, all for the difference of a single mil, which isn't important for your design anyway.

PCB fabricators enforce design rules so they can be confident that they'll be able to produce your board to meet their quality, yield, and time standards. It's really important to set up the design rules in your CAD software to mirror those of the fabricator you'll be using. These design rules usually appear in a table on the manufacturer's website. If your manufacturer has a DRC or design constraint file for your CAD package, download it and use it. This will automatically import the minimum feature sizes and spacing requirements for the particular process you'll be using. However, a manufacturer's DRC file may not always be complete, so take the time to check it against the rules listed on the manufacturer's website.

Some manufacturers even have dedicated tools you can use to check your Gerbers before you submit them. For example, AdvancedPCB has a website called FreeDFM.com that allows you to upload your Gerbers and automatically look for and identify errors before you submit them for fabrication. While FreeDFM.com is only set up to check against AdvancedPCB's rules, those rules are fairly standard, so it can still be useful for finding mistakes even if you don't ultimately fabricate with Advanced Circuits.

If you'll be designing a board with high-speed or RF requirements, look at the available stackups and materials of your PCB manufacturer before designing the impedance-controlled structures. Knowing what stackups are available can significantly impact your cost. Exotic or custom stackups are often much more expensive than the standard stackups a board house may use. Also, they often have much longer lead times. In many cases, adapting to the available stackup isn't very difficult and may change your layout only slightly, but those small changes can save you thousands of dollars and weeks of time.

Be aware that many manufacturers consider a hole and a via two different things. They usually require a larger clearance around a hole than a via. If you accidentally use too small a clearance around a hole, the manufacturer will usually edit your design to add extra space if it looks like it won't affect the design. But it's better if you catch the problem before they do.

PCB Markings

All copper needs to be pulled back by at least 1 mm from the edge of the PCB or from any edge that will be cut or milled out. PCB fabricators have different requirements for this, but 1 mm should be enough to meet the requirements of most fabricators. This is also important if you'll be separating boards yourself with a band saw or Dremel. The reasons are the same: If there's copper on multiple layers along the line where a cut will be made, the copper can get smeared down along the edge during the cut and short to other layers.

To improve silkscreen resolution (with some cost increase), use two different layers of different color solder mask instead of standard silkscreen. You'll need to talk to your PCB fabricator about this, but solder mask is placed at much tighter tolerances and has sharper edges than normal silk-screen. This is the technique that Arduino uses, for example, to achieve the fine visual detail that makes their PCBs look great.

If your board has silkscreen labels, set the font size small enough so that component designators fit next to their component, but not so small that the printer cannot clearly print it. Make sure all designators

and text can be read in one of two directions (see Figure 7-1). This will keep you from having to rotate the board constantly when trying to read the silkscreen.

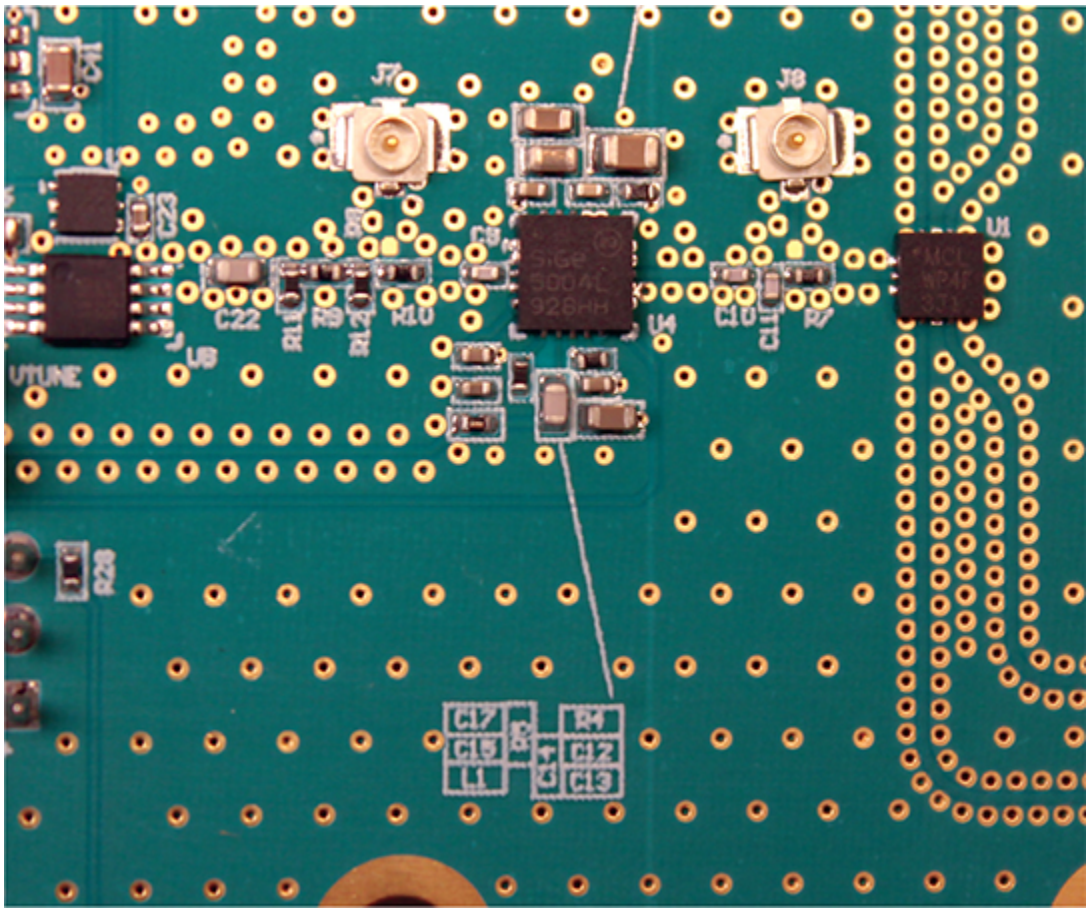


Figure 7-1: Keep the silkscreen rotated in one of two directions so it's easy to read without spinning the board around too much.

If you can't fit designator labels because component placement is too dense, you can place the designators nearby with the same orientation as the components. You can see this on the right in Figure 7-1 with designators C17, C15, C4, R4, and their neighbors. It also illustrates how you should keep silkscreen off vias and pads so that it doesn't get cut off. Most PCB fabricators will remove all silkscreen that overlaps pads during their DRC process, but you should check it yourself before you send it out. Silkscreen that overlaps an untented via can be difficult to read, as shown in Figure 7-2.

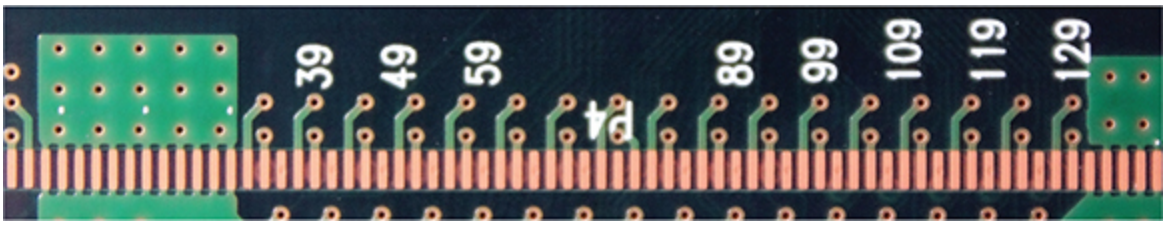


Figure 7-2: The silkscreen in the middle of this image is over the top of a via, making it difficult to read.

Additionally, keep the pin 1 marker out from underneath the component for the same reason. Some people like to use a square with a corner cut off to outline where the chip sits in the footprint and designate where pin 1 is. You can do that, but it's better if you also use a silkscreen dot next to pin 1 so that it's still visible after assembly. For even more clarity, you can use one or two small triangular arrows to point to pin 1.

Be sure to label the polarity of any polarized parts like electrolytic capacitors and diodes. Through-hole electrolytic capacitor footprints often have a solid area of silkscreen underneath the negative terminal and a plus sign next to the positive terminal outside the footprint. Diodes ought to have a polarity marker that isn't underneath the footprint so you can inspect it after assembly. Likewise, if your board has any replaceable parts like fuses, mark the type and rating of the component in silkscreen next to its location on the PCB so it's easy to know what replacement to use. It's also a good idea to put the PCB version and date in silkscreen so you don't confuse multiple iterations of the board, as shown in Figure 7-3.

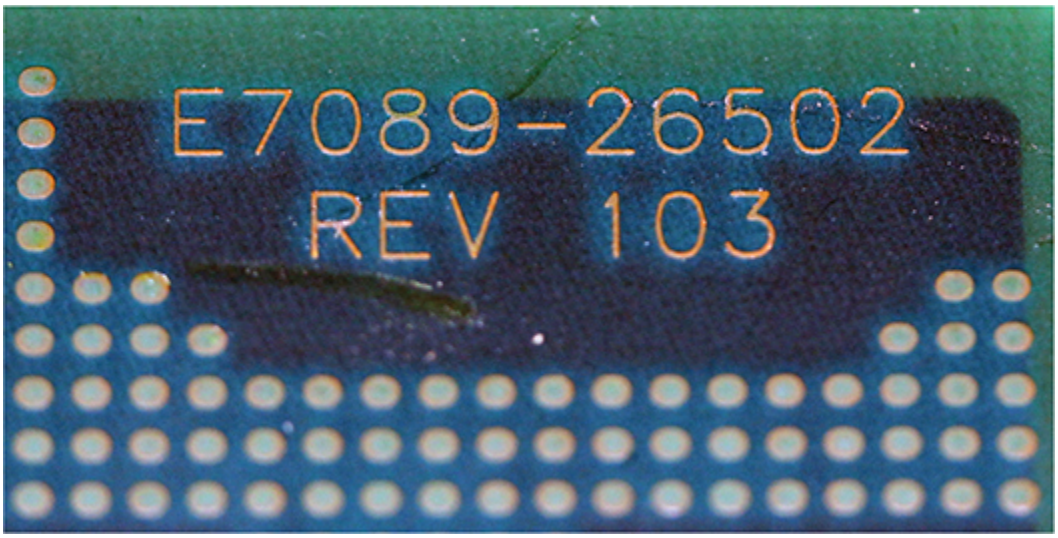


Figure 7-3: A version number and other markers for a PCB revision. In this case, they were put on a copper layer instead of in silkscreen.

Keep in mind that PCB fabricators will usually add some markers to your board for their own reference, as in Figure 7-4. You can specify a side or location for them if you want.

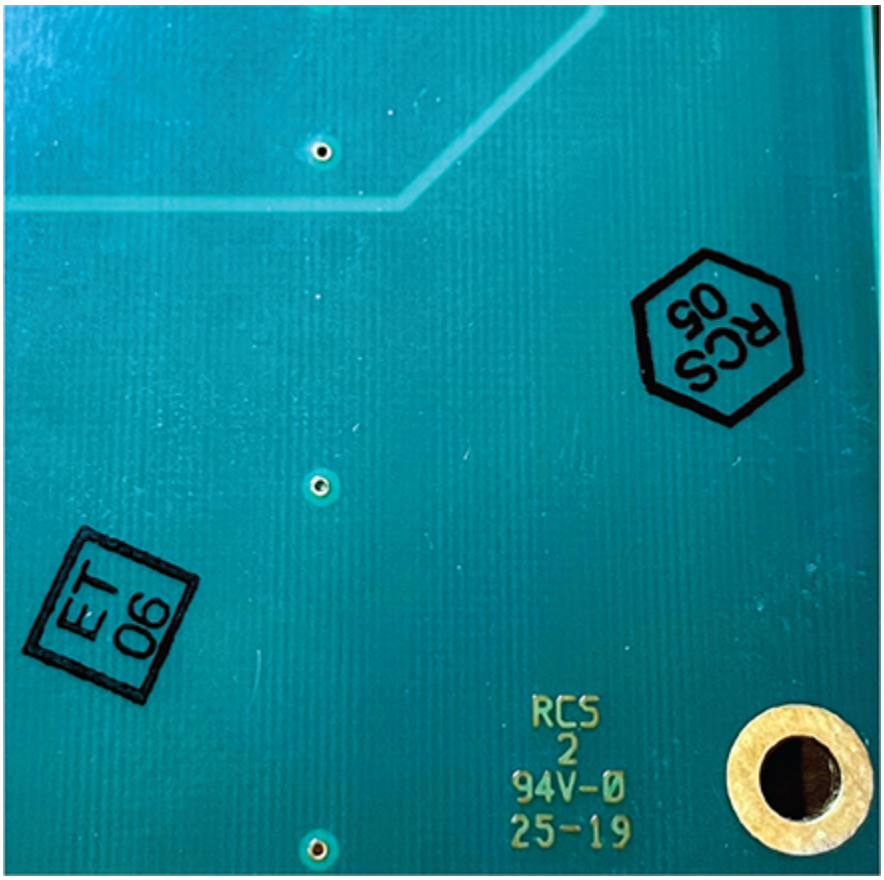


Figure 7-4: Markings added by a PCB manufacturer

All the markings in Figure 7-4 were added by the PCB manufacturer for their own tracking purposes. The "ET 06" stamp probably refers to a passing electrical test from inspector 06. "RCS 05" may refer to inspector 05 at the manufacturing company (RCS is an abbreviation of the company name). The markings in copper at the bottom of the image include the company name, a date code (25th week of 2019), and "94V-0" to indicate that the stackup is UL 94V compliant (a flammability rating).

For your own markings, it can be helpful to add a small rectangle of silkscreen in a corner of the board somewhere, as in Figure 7-5.

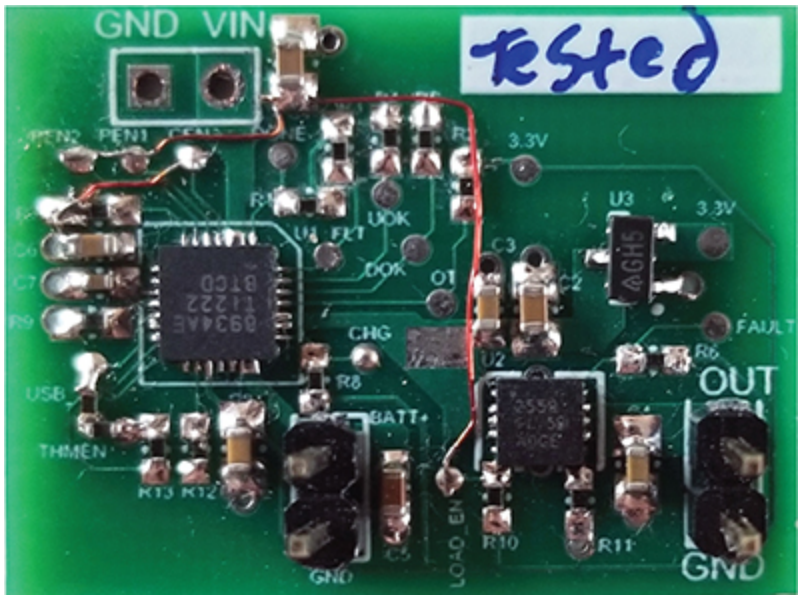


Figure 7-5: The rectangle of silkscreen in the top-right corner provides space for a label.

You can use the silkscreen rectangle to write notes or label the PCB with a marker. For example, you might note if a board has been tested or add its serial number.

Copper Thieving

Sometimes you'll see PCBs that use a technique called *copper thieving*, where unconnected areas of copper are added to a board. Figure 7-6 shows an example.

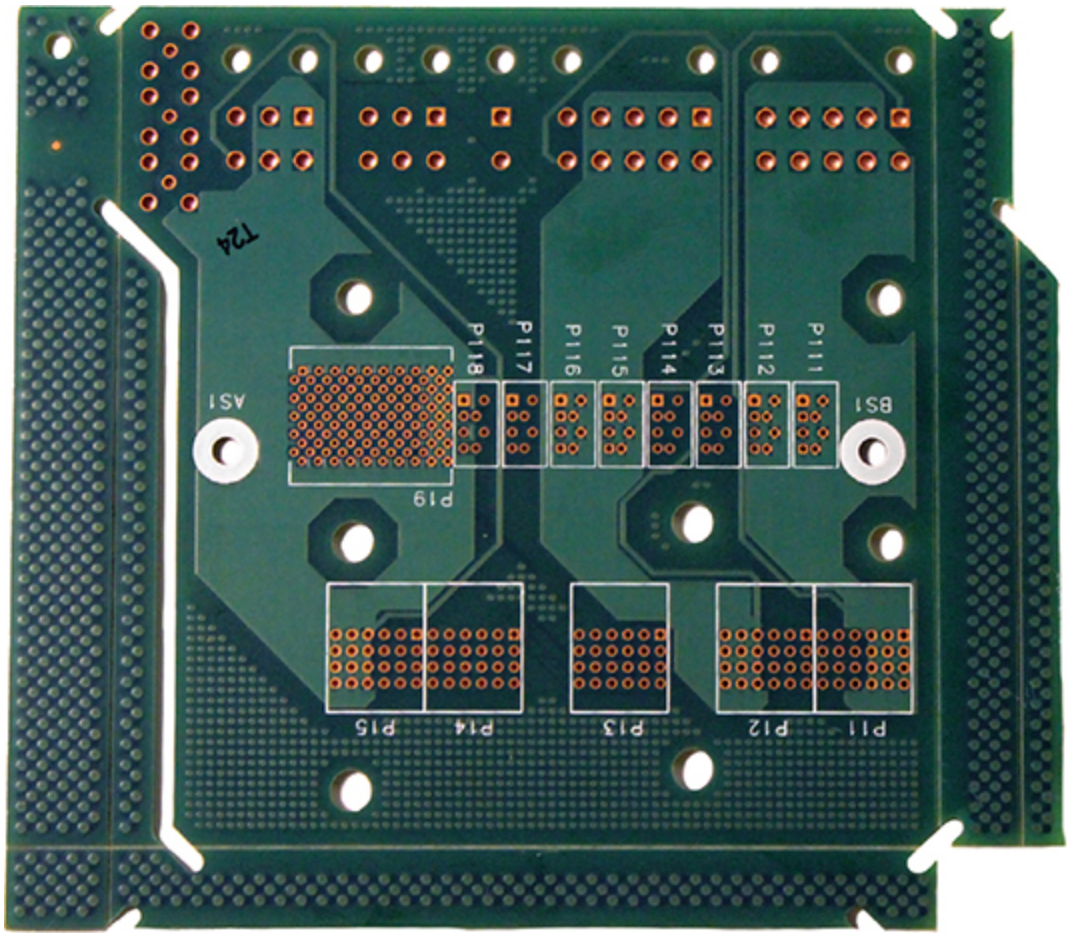


Figure 7-6: The checkerboard pattern on the perimeter of this board is an example of copper thieving.

Copper thieving is done because major problems can occur if there's significantly uneven copper distribution across the PCB. For one, copper will be laid down unevenly during the electroplating step of manufacturing, which will cause uneven plating of holes and vias as well as an inconsistent outer-layer copper thickness. Additionally, the uneven copper can make the board twist or warp.

The best way to avoid these problems is to simply flood-fill your signal layers with ground. This will fill all the remaining spaces between signal routes with copper and prevent any copper distribution–related fabrication problems. If you aren't able to do this for some reason, the fabrication will have to add thief copper either on your PCB or right next to it on the panel. This can decrease the number of PCBs that you can get on a panel, which will increase your cost.

Panelization

Circuit boards aren't made one at a time. Multiple copies are arranged onto a single large PCB that gets processed all at once. The art of arranging those copies in an efficient yet easy-to-manufacture way is called *panelization*, and separating the panel into individual boards is called *depanelization*.

A *fabrication panel* is typically 18 inches by 24 inches, although fabricators often have other sizes as well. Fabrication panels are used to create the blank PCBs through etching and drilling processes. After the blank boards are done, the fabrication panels are often cut into smaller panels called *assembly panels*, which are loaded into the machines that place and solder the components on the boards. You can see an example of an assembly panel in Figure 7-7.

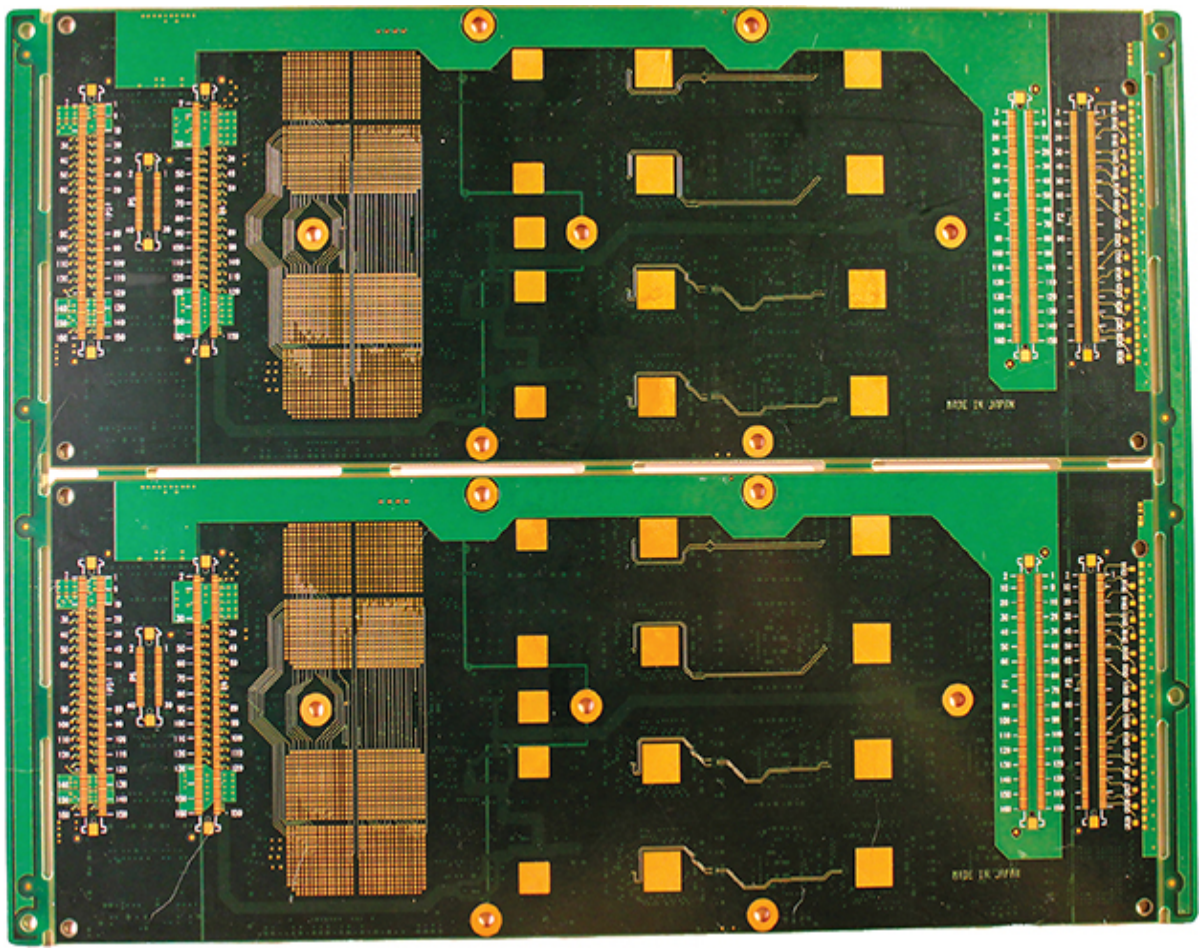


Figure 7-7: A PCB assembly panel. This panel has two PCBs on it and is supported on the left and right sides by the conveyor.

If you're making a small number of PCBs or are prototyping something, don't spend time trying to manually panelize your design. The fabricator will take care of that for you, and they'll do a better job of it. You only need to worry about designing PCB fabrication and assembly panels yourself when you're dealing with high volumes in production.

If you *are* building production boards, you'll want to fit as many boards as possible on each panel to get the highest yield. There's dedicated software to help plan this out, but you should also work with your production engineer before trying to run any large number of boards through the assembly process. It's not as simple as just cramming as many boards on there as possible, since adding tabs and perforations to separate the boards later can cause problems during assembly if

they're not placed well. Boards can sag when solder paste is applied, for example, or they can vibrate and flex in the pick-and-place machine. In general, a well-designed PCB panel has about 70 percent usage. The rest ends up as scrap from spacing and clearances between individual PCBs and the edge of the panel. Board houses will request 100 mils between the edges of each individual PCB in the panel, as well as a 0.5-inch edge around a two-layer panel and 1 inch around a multilayer panel.

If you want your design to be compatible with standard assembly equipment, the smallest outer dimension of your PCB should be at least 2 inches. Typical assembly line rails don't adjust to any smaller than that. You have two options if your design is less than 2 inches on a side. You can either add additional copies of your design to the panel until the overall panel is big enough, or you can add processing edges that will be scrapped after assembly. This technique also applies to PCBs that have an unusual shape (for example, circular PCBs) and can't easily slide down the two rails of the assembly line.

When you panelize a design, use V-grooving to make it easier to separate boards. This is a process that some PCB fabricators offer where they cut a V-shaped groove on the top and bottom of the PCB outline, as shown in Figure 7-8. The individual PCBs stay in the panel, but the grooves allow you to use your hands to snap apart the PCBs into individual boards. This has the advantage of requiring very little force and leaving a relatively clean edge without using a band saw.

Figure 7-8: An edge-on view of a V-grooved PCB

You may want to perform basic testing after the PCBs in the panel are assembled but still panelized. Once testing is complete, you can either route out the boards with a mill or, if the panel is V-grooved, just break them off by hand. This will let you avoid the vibration and dust created by the milling process.

If you plan on using V-grooving on a design that will go in an enclosure, be aware that the V-grooving may leave a few extra mils on all sides of the board. Also, V-grooving machines can't stop and turn in the middle of the board like a mill can, so you have to make all V-groove lines go all the way through the board, edge to edge.

Another limitation of V-grooving is that it can't be used on very thin boards; usually a thickness of greater than 3 mm is needed. Also, because the tool that cuts the groove is V-shaped, you need to keep tall components away from the cut line. Your PCB manufacturer can give you exact numbers based on their machines, but an example guideline might be to keep components taller than 25 mm at least 5 mm away from the cutting line.

If your PCB isn't a rectangle, you can use a technique like the one in Figure 7-9 to fit it in a rectangular panel while making it easy to depanelize by making only straight-line breaks.

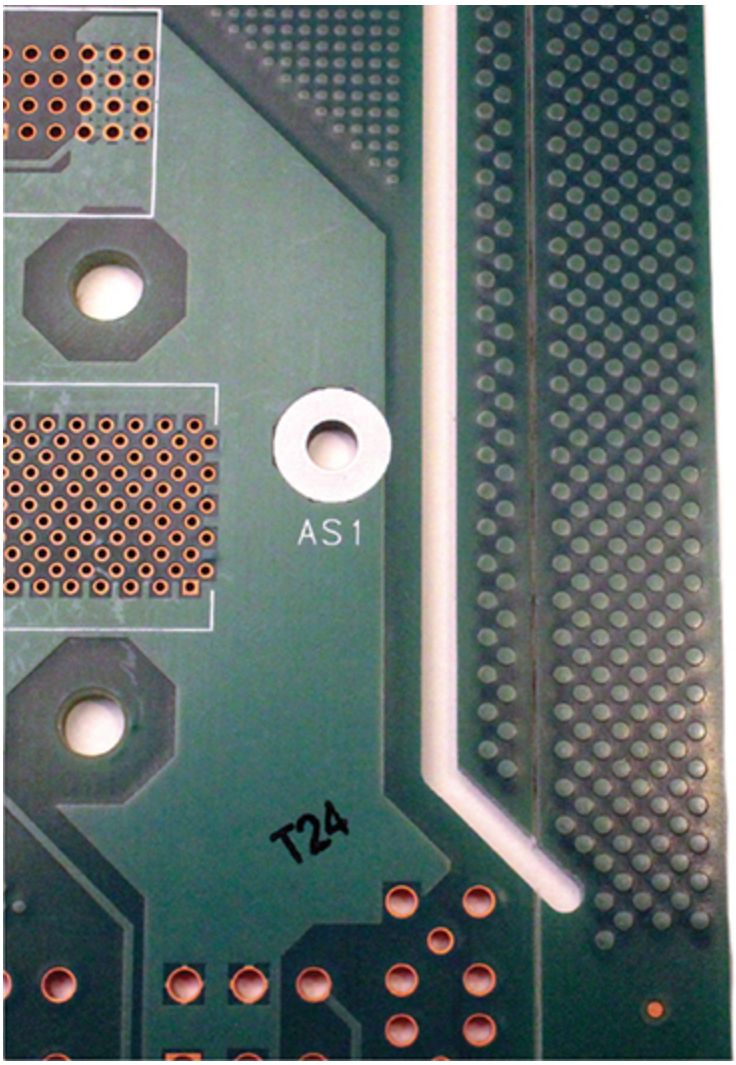


Figure 7-9: Example of using a routed slot with a V-groove for nonrectangular PCB depanelization

The thin white line to the right of the slot in the figure is a V-groove. The routed slot to the left of it will create a contoured board outline when the V-groove is broken.

There are a couple of options besides V-grooving for depanelizing individual boards from a panel. Solid tabs are mechanically robust, but they have to be routed out with a mill or laser. Perforated tabs are solid tabs with small holes in them to create a perforation. These are sometimes also called *breakaway tabs* or *mouse bites*. You can use a pair of strong angle cutters to cut along the perforation, or you can break it with your hands. To remove a perforated tab, use a pair of pliers to grip the edge. Bend upward until you hear a crack and then stop. Now bend

it back downward and remove the edge. This will prevent the PCB from delaminating.

You need to keep all components and copper at least 100 mils from the perforation holes. You also need to take special care with SMT MLCC capacitors. Because these capacitors are very inelastic, the force of bending and snapping off the tabs can actually cause microscopic cracks through the middle of the capacitor and cause it to fail immediately or much later, when moisture has seeped into the middle of the capacitor. For that reason, you should place MLCC capacitors about twice as far away from the perforation as everything else (about 200 mils).

Typically, a perforated tab will have five holes in it. You can use fewer if you need to, but use at least three holes. The fewer the holes, the closer together they should be to maintain board strength. IPC-7351 has detailed information on spacing and clearance for perforated tabs. Because you want smooth edges on your boards once they're depanelized, you need to make sure the perforation holes are along what will be the final edge of the board. Don't place perforation holes in the middle of the tab or on the outer edge of the tab, or else there will be a little nub of PCB material sticking out after you've separated each board.

If your design is going to be wave-soldered rather than reflowed, you need to make sure there aren't any large holes or gaps in the panel. If there are, solder from the solder wave can lap up over the edge and spill onto the top side of the board. Any hole or space that's larger than 600 square mils needs to have a *knockout*, a section of PCB with no components or traces on it that will be scrapped after depanelization. While the 600-square-mil rule isn't required for designs that will be reflowed, you can still use this technique to prevent flexing and vibration during assembly if there are any parts of the panel that aren't well supported or anchored to the panel edge.

The crudest way to separate boards is to use a hacksaw or band saw. You'll need to use this technique if you're building only a couple boards and don't have any of the tools needed for depaneling or don't want to pay for V-grooving. You'll also be limited to this technique if you're

using *stealth panelization*, which is when you put multiple designs on a single board for a fabricator that doesn't allow panelization. Some quick-turn places don't like it when you try to include multiple designs on a single PCB, but as long as you don't use tabs or V-grooving, you can usually sneak it by them.

Since sawing (especially with a hacksaw) is fairly crude, you need to make a channel at least 5 mm wide that contains no copper on any layer for the path of the saw. If you have a board with multiple layers and flood fills on any two of those layers, sawing through those layers can smear copper from one layer to another along the cutting edge. This can cause shorts.

Don't forget that your panel will also need tooling holes. These are holes that will be used for test jigs (such as a bed-of-nails jig) to align the board to the fixture and for automated through-hole assembly equipment. Tooling holes are usually 125 mils in diameter and are unplated. You need at least three tooling holes, and they should be about 200 mils from the edge of the panel.

Design for Assembly

Making it easy to solder your components on your PCB will ensure that your product can be produced in large quantities with high yield, which will ultimately save you money. Making a board easy to assemble also tends to make it easy to rework, which is important both for quickly making changes while prototyping and for repairing and refurbishing devices in production. In this section we'll look at tips that apply to both robotic and hand assembly.

Fiducials

If your design is going to be populated with a pick-and-place machine, you'll need to include global and local *fiducials*. These are small visual markers on a PCB that the pick-and-place machine's camera can use to identify a PCB's orientation and coordinate plane. Fiducials are typically round or square features on the top layer of copper, such as a 35-mil diameter circular pad with a 100-mil diameter solder mask relief. The

copper layer is used rather than silkscreen because PCB fabricators are able to place copper features much more accurately.

There are usually two or three global fiducials on a PCB, depending on the size and shape of the board or panel. Talk to your production engineer to determine exactly how many fiducials to place and where. You may want to add two extra local fiducials next to any large chip (like an FPGA or processor) to ensure accurate placement and alignment of that chip. Figure 7-10 shows an example.

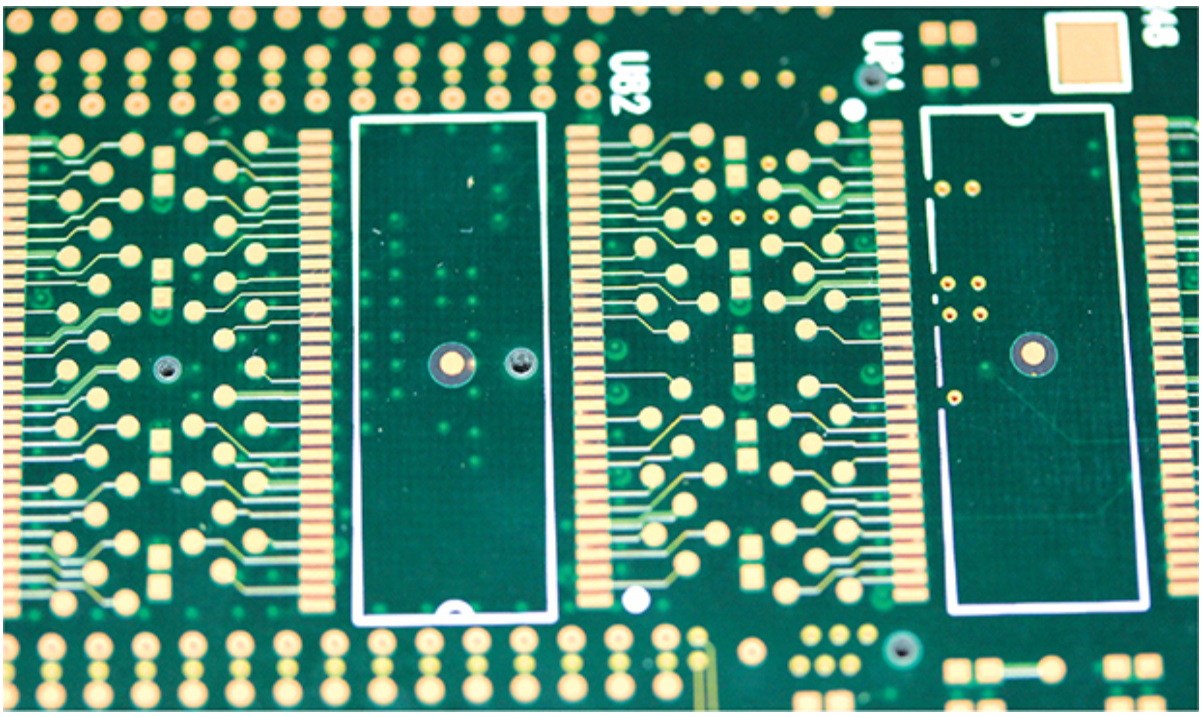


Figure 7-10: Each chip footprint shown here has a fiducial in the middle to help the pick-and-place machine line up the part correctly. This is sometimes done for parts like these with a large number of pins.

You should place fiducials asymmetrically so that the pick-and-place machine can resolve any rotation of the board uniquely. The fiducials should also be at least 150 mils away from the edge of the board so they are not obscured by the rails of the conveyor during assembly. When panelizing a design, put fiducials on the panel itself as well as on individual PCBs.

Including fiducials isn't the only way to ensure the pick-and-place machine works properly. You should also make sure all SMT component

footprints have their coordinate origin located in the middle of the part and that all through-hole components have their coordinate origins on pin 1. If you export your pick-and-place instruction file from your CAD program and the origins of footprints are offset from their conventional locations, the pick-and-place may incorrectly place your part with the same offset. Figure 7-11 shows examples of an SMT component footprint with the origin placed correctly and incorrectly.

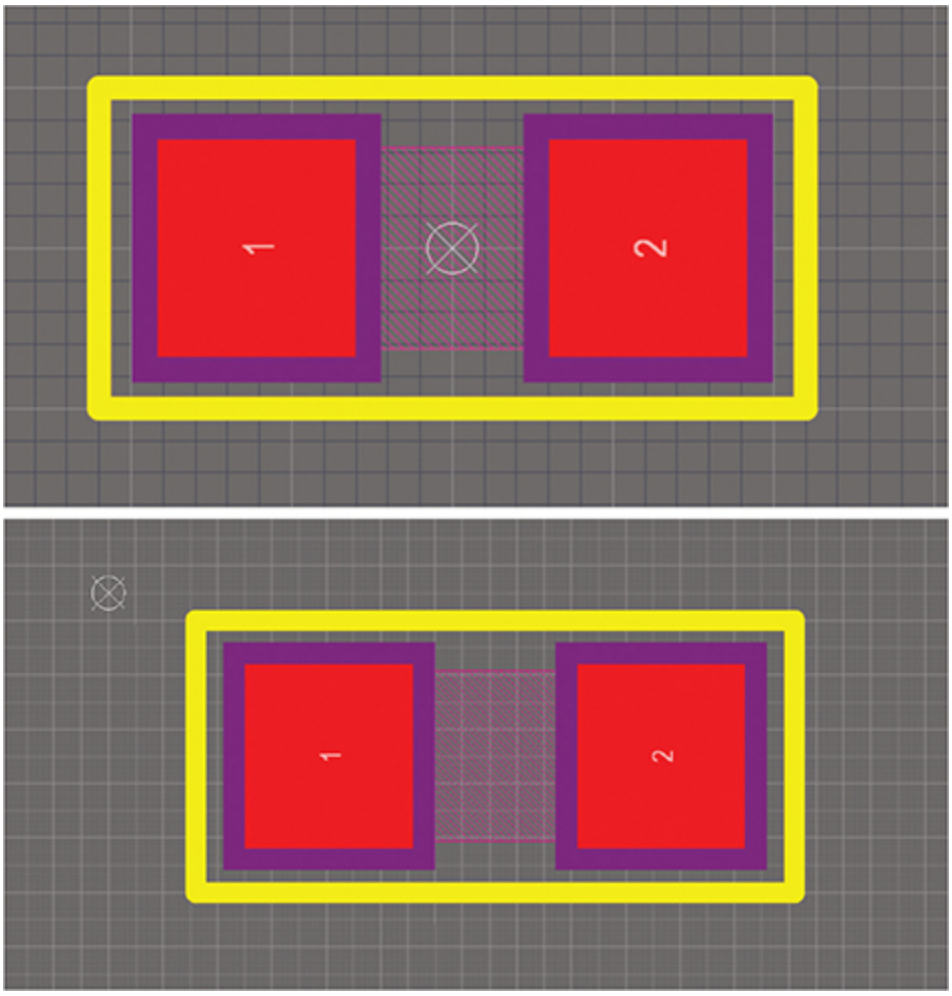


Figure 7-11: A component with the origin placed correctly in the middle of the part (top) and incorrectly off to the top left of the part (bottom)

Good assemblers will always manually check component coordinate locations and make corrections, but you should still try to get it right yourself.

Markings to Aid Assembly

If there's space for it, all notes, component designators, and pin markers in silkscreen should still be visible after all components have been populated. If your CAD tool supports 3D component models, you can use that to assist in determining whether any silkscreen will be occluded.

Put a silkscreen dot or triangular arrow next to pin 1 of every chip. This makes it easier to trace out signals, replace parts, and check that the part was populated in the right orientation. You can also add small silkscreen tick marks next to every fifth or tenth pin on large connectors and ICs to make it easier to count pins during debugging and bring up. Another option is to mark the pin numbers at each corner of the chip.

Two-pin discrete parts like resistors, inductors, and capacitors can often be difficult to differentiate during manual assembly if they're packed close together. To help tell which pad goes with which footprint, you can either use a rectangular silkscreen outline around the footprint (as shown in Figure 7-12, top left), or use two small silkscreen lines to connect the two pads (Figure 7-12, top right). If your footprint is 0805 or larger, you can even connect the two pads with a small silkscreen symbol of the component that gets populated there (Figure 7-12, bottom). You shouldn't do this for parts smaller than 0805, since the silkscreen can sit proud of the surface and lift the component enough to cause it to rotate out of alignment during reflow or handling.

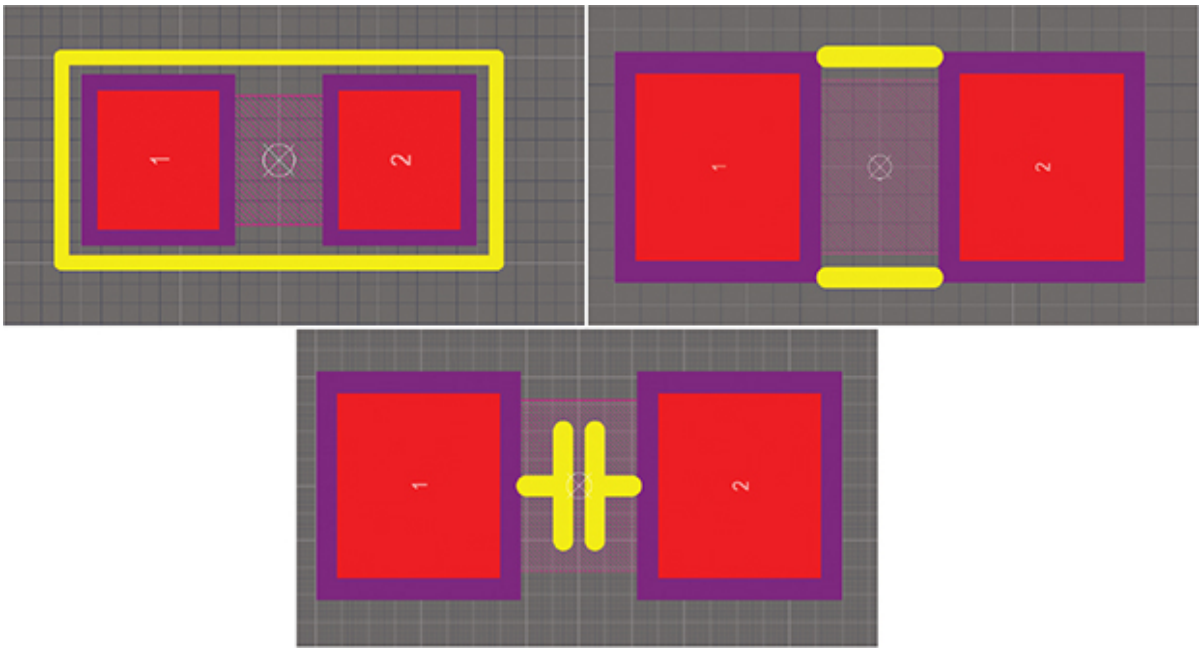


Figure 7-12: Setting off a two-pin component footprint using a silkscreen outline (top left), connecting lines (top right), or a component symbol (bottom).

Drawing a silkscreen border around each part will also help make sure components and connectors aren't going to hit each other, the enclosure, or any other mechanical features. The borders let you immediately see how much clearance you need to leave around each component when you're placing parts. You can also put these borders on a separate layer in your CAD software, typically called a "courtyard," and set a DRC rule to alert you to any collisions. An even better method, if your CAD package supports it, is to use 3D models to ensure nothing collides. Many component manufacturers provide free 3D models of their products that you can import so you don't need to painstakingly model every component you use from scratch.

Soldering Considerations

Don't put untented vias too close to the pad of an IC, especially if those un-tented vias are underneath the IC. Solder can bridge the pad and the untented via, causing a short that's completely invisible without removing the chip. I've seen this happen even on TQFP parts. There will be no externally visible shorts on the legs themselves, but solder can wick underneath the chip and cause an invisible short there. The best

way to avoid this problem is to move the vias farther away from the chip pads, out from underneath the chip if possible, or to just tent the vias.

If you're hand-assembling a PCB, you can do yourself a favor by lengthening the pads slightly so they're easier to access with a soldering iron. This applies only to footprints that you'll actually be soldering with an iron, so packages that don't have externally accessible pads and require reflow should use the footprint dimensions recommended in the datasheet. You can use this trick to allow you to solder QFN parts by hand: You can drag-solder the pins when the pads project out underneath the part. This creates a solder fillet between the pad and the upper edge of the pin. Some QFN chips are sold specifically for this; they're called *side-wettable* QFNs. They're great because they don't require X-ray inspection.

Some chips have externally accessible pins but with a ground or thermal pad underneath. If you need to solder a chip like that without reflowing it, you can put a large plated through hole in the middle of the ground or thermal pad. That way, you can solder the chip's external pins on the top side and then flip the board over and put your soldering iron in the hole to make a good connection to the plated hole and ground plane.

Isolation between parts and traces is very important when manufacturing a PCB without solder mask. If you're using a circuit mill, pads and traces tend to peel more easily, especially as you add heat to solder. If you have to add more heat to overcome more thermal mass, you'll have a higher risk of peeling up an adjacent trace or damaging the board. Physical spacing and isolation will help prevent this.

Another thing that makes a board with no solder mask more difficult to work with is the fact that it's much easier to accidentally short pins, traces, and planes together. This is especially true when using a soldering iron instead of solder paste and hot air. The farther apart you can get your traces apart from each other after they leave the package pins, the lower the chance you'll have of getting a solder blob that shorts out three or four pins at once. Using flux is highly recommended when assembling these "barebones" boards.

Enclosures

Think about how your PCB will be supported within its enclosure. Screw bosses and standoffs are popular methods for locating and fixing a PCB within an enclosure. Also think about how your device will be used. For example, a bathroom scale will have hundreds of pounds exerted on it. Will your PCB bend when that happens? In complex cases like this, you can use finite element analysis to model the result of arbitrary forces on your enclosure and PCB. You should work closely with your mechanical engineering team on these issues *before* the design is finalized.

To be clear, PCBs should never be bent or flexed in a product. Little to no force should be exerted on the board; it must be fully supported mechanically with no "diving boards" or similar structures. This will help you pass shock/vibration testing and prevent long-term reliability problems.

If your enclosure needs to be grounded, the mounting holes on your PCB need to be plated and connected to ground. They also need to have an annular ring with exposed copper around the hole so the bottom of the screw head will make solid contact with the ground plane. A ring of vias on the annular ring can help stiffen the structure. All traces and components should be at least 100 mils from the edge of the annular ring. Otherwise, if the screw is overtightened and begins to bite into the PCB, it can short or open traces. Star washers should never be used on a PCB. They deform the substrate, which causes them to stop applying pressure, and then over time the screw can back out. Use a screw with a split-lock washer instead.

Screw holes need to be accessible to a screwdriver without having to disassemble anything. This applies more to the mechanical design of how the PCB is attached to the enclosure, but don't "paint yourself into a corner" and end up with a design that's impossible to assemble. The best way to avoid this is to make sure that no fastener heads are covered up by anything else. If that's not possible, then think through the order in which you'll assemble and disassemble the device when choosing the fastener locations.

When you're drawing or importing 3D models for each component, always use the worst-case tolerance. For example, if your enclosure height is 6 mm off the surface of the PCB and a part's datasheet says that it's 5 mm +/– 0.5 mm tall, model it as 5.5 mm tall, not 5 mm. The same thing holds for the other direction. If you need a part to be at least a certain height, model it as the worst-case tolerance in the other direction (so 4.5 mm in this example). Tolerances add up, and you want to make sure you can handle the worst case.

Consider making the PCB's outline asymmetrical. This helps assemblers because there's only one way for the PCB to go into the enclosure. For example, you can cut off one corner of a square board to make the orientation clear. However, the mechanical design must also reflect this. If you have a square groove that the PCB sits in, it doesn't matter if the PCB has had a corner removed to make it asymmetrical because it will still fit in that square groove in any orientation. The groove also needs a corner removed.

Cables and Connectors

If you're hand-assembling a board, connectors need to have enough clearance around them so they can solder them without melting the plastic connector housing. They also need to have enough clearance for the mating connector, including the wires that stick out from the back of the mating connector (if it's a cable). The first centimeter behind a connector on a cable can be surprisingly rigid, and there needs to be room for those wires to bend around on their way to their destination.

Route cables cleanly through your design and plan where they will go. Take into account their thickness, the number of wires, and their rigidity (stranded vs. solid core). You can create cable bundles using zip ties, a cable sheath, or lacing, an older technique that uses string to keep wire harnesses neatly bundled. You can also build cable channels directly into your enclosure that will help guide wires and reduce their chances of being pinched by something during final assembly. This is the best technique for assemblers, since it's clear exactly where the wires need to go. You can use tape or adhesive to keep the bundles from falling out of their channels.

If you have two connectors next to each other, make sure there's enough room for both of the mating cables to be plugged in at once. Some cables have a very wide lip around their connector, and this can vary quite a bit even within the same connector type. The plastic shell around the connector can easily take up double the space of the connector itself, both horizontally and vertically. Think about which cables you're going to use, and make sure there's plenty of space between adjacent connectors. It can help to show the connector dimensions with a silkscreen outline, as in Figure 7-13.

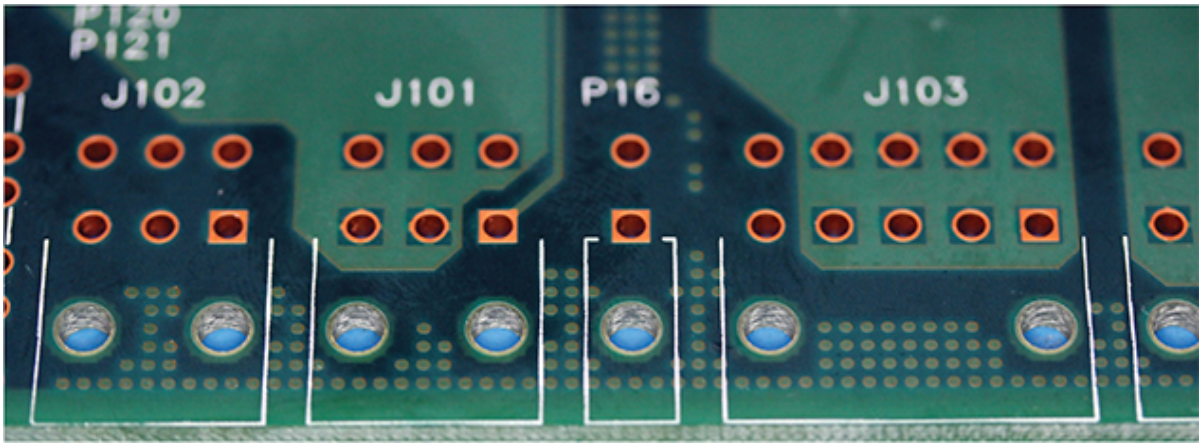


Figure 7-13: Use silkscreen outlines of the connector dimensions to ensure you leave enough space between adjacent connectors.

You can use the same silkscreen connector outline technique for non-edge connectors to check for clearance and to illustrate the correct orientation of the connector for assemblers. Figure 7-14 shows some examples.

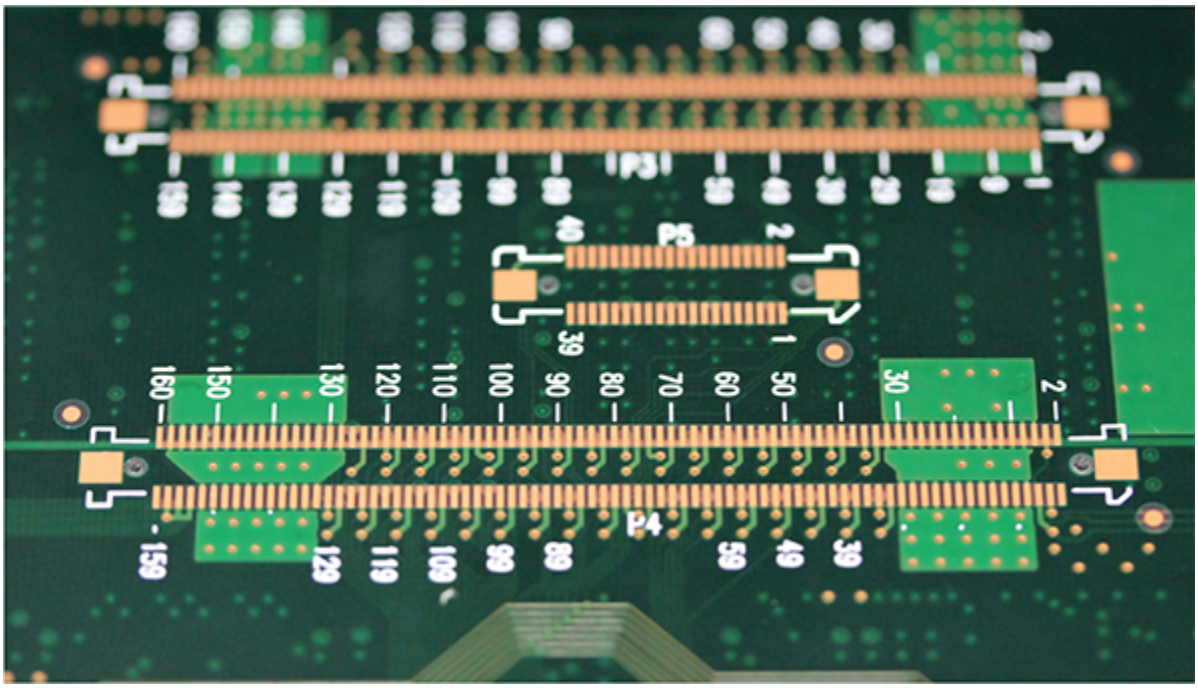


Figure 7-14: Outlining non-edge connectors with silkscreen. Pin numbers have also been included here to assist in debugging and bring-up, as well as to help prevent connector orientation mistakes.

Besides connectors, mind the clearance around switches and dials as well. For example, two potentiometers may have enough clearance next to each other to turn when there aren't knobs on them, but depending on the size of the knobs, it can easily become impossible to have both on at once without them rubbing against each other. Plan the finished knob locations first and then figure out where the potentiometers need to go afterward, not the other way around.

Ribbon cables and flat flex connectors (FFC) need to have enough clearance around them to access the latching mechanism or tabs used to insert and remove the cable. Avoid having to use a tool to connect or remove the cable if possible.

No matter what kind of cables you use, think about the order in which you'll assemble the device and how that will affect cable placement. For example, if you have an FFC underneath an overhang that must be assembled first, it might be impossible to plug the cable in. The length of the cables needs to be taken into account, too. Think about how much slack you'll need to make the connection when the

device is pulled apart, but also think about where that extra slack will go once the enclosure is buttoned up.

If there are lots of cables inside your device, and especially if they're FFC or thin ribbon cables, consider rounding your board's outline to prevent pinching or otherwise damaging the cables during assembly. Everything may look fine when it's apart, but tightening the final screws on the outside can apply pressure and cut or damage cables that have been poorly placed.

Design for Test

Usually after a board has been tested, a serial number is generated and sometimes affixed with a sticker. You can use silkscreen to show test personnel where to affix the sticker, as in Figure 7-15.

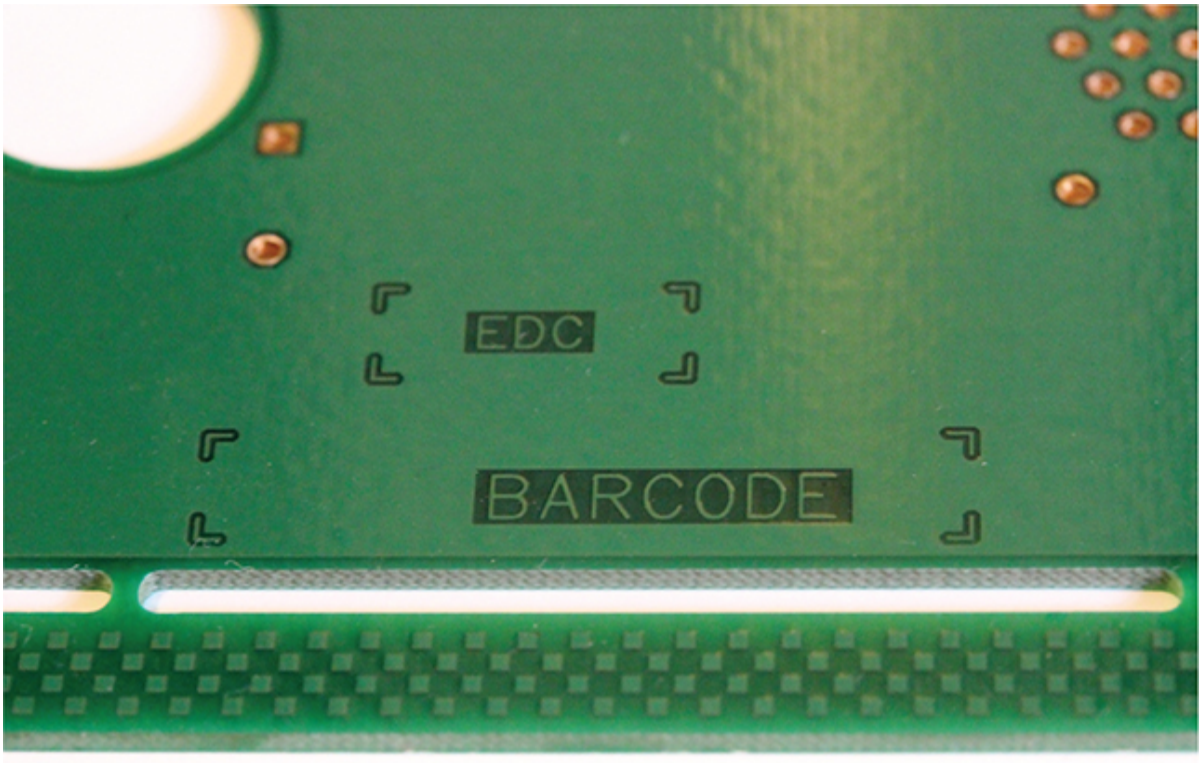


Figure 7-15: Use a silkscreen outline to show assemblers where to put identifiers like a barcode for tracking individual units.

Differently shaped annular rings around vias can signify connections to different planes. For example, you can use an untented octagon

rather than a circle to indicate a power plane. This makes identification easier during debugging and testing.

If any parts need to be accessed by a person, tool, or test fixture (for example, a potentiometer, connector, jumper, button, or test point), make sure there's enough clearance around the component for the tool, fixture, or finger to access it. If someone's finger is going to be used, make sure it's safe to put a finger there. That means no nearby high voltage or hot parts that you could accidentally touch. You should be able to get a soldering iron and tweezers to each component. If a part has a screw on it (like a TO-220 package), make sure there's enough room to get a screwdriver to the screw without hitting anything.

It's not practical to add test pads to every single net, but you can use vias as test pads as long as they're not tented. Consider leaving vias untented for your design's first prototype so it's easy to use needle probes to connect to any trace with a via. You can even solder wires or add discrete components directly to the exposed vias if rework is necessary. If you need to probe a net with a test fixture for production, you'll need to use a real test point. A via is too small and fragile, and your production board should use tented vias to prevent accidental shorting.

Test pads should all be on the same side of the PCB if possible. That makes the test fixture much easier to build. Additionally, if all of your components are on the same side of the board, you can put all of your test points on the *back* of the board, making it very easy to build a bed-of-nails test fixture without having to worry about hitting any components. To make sure your test fixture pins can hit the test pads, your test points need to have specific dimensions. The website *https://www.qatech.com* has great references to help you figure out what dimensions to use.

If at all possible, align all of your test pads to the same grid size. A lot of people like to use a spacing of 100 mils. This allows you to use 100-mil header pins, breadboards, perf boards, and many readily available connectors to assist you in building jigs for programming and testing.

Designing an RF circuit to be easy to test can be challenging because extra components and pads can affect the impedance of transmission lines. One method is to use a *coupled line coupler*. This is a little strip that runs parallel to the transmission line you want to measure and will AC couple some of the signal. That will allow you to measure a lower-amplitude version of the signal without using any components or interfering with the transmission line. You can use online calculators to figure out the length and separation distance for a coupled line coupler, or you can simulate it with a tool like Keysight ADS.

Another way to make it easier to test RF circuits is to use one of a family of special RF connectors made by Murata. These are tiny surface-mount connectors that you can put in series with a transmission line and that act like a short when not connected to anything. When you plug in a cable, the connector has a tiny switch that actuates and connects the input of the connector to the cable and disconnects the output of the connector. These connectors work mostly up to 6 GHz, but some are rated for as high as 12 GHz. Insertion loss is also generally pretty low, about 0.2 dB, but some versions are as high as 1.8 dB. For a full list of these connectors, look up Murata Microwave Coaxial Connectors, or look up part number MM8130-2600B, to start.

Design for Reliability

If your product will be mass produced, think about how quality will be controlled for reliability. Inspection and testing are the two most common ways to verify quality, and both of those techniques require thought during the design phase. Are there critical parts of the design that need to be visible to inspectors? Components are usually inspected during assembly so that subassemblies can be inspected before they're integrated into the rest of the design. Think about how your device can be systematically tested and inspected during assembly so that nothing needs to be unscrewed and the right parts are tested in the right order.

To ensure reliability, use the same PCB manufacturer for both your prototypes and the final production PCB if possible. Absolutely do not

build production PCBs with a manufacturer without building any prototypes with them first, even if you've proven the design with a different fabricator. Not all PCB manufacturers are created equal. Inspect the prototypes carefully: Pay attention to how close via holes are to the middle of the pad, look for any smeared copper or shorting, and check how evenly the solder mask is applied. If possible, verify that the manufacturer is actually meeting the IPC class that they claim they're hitting. Some factories (especially in China) have been known to lie about being IPC certified or will show you an expired certification.

One potential reliability issue that's often overlooked is *tin whiskers*, tiny filaments (0.5 to 10 microns thick and up to 1 centimeter long) that grow from tin-coated surfaces. Materials other than tin, like zinc and cadmium, can also grow whiskers. Figure 7-16 shows an example of a tin whisker.

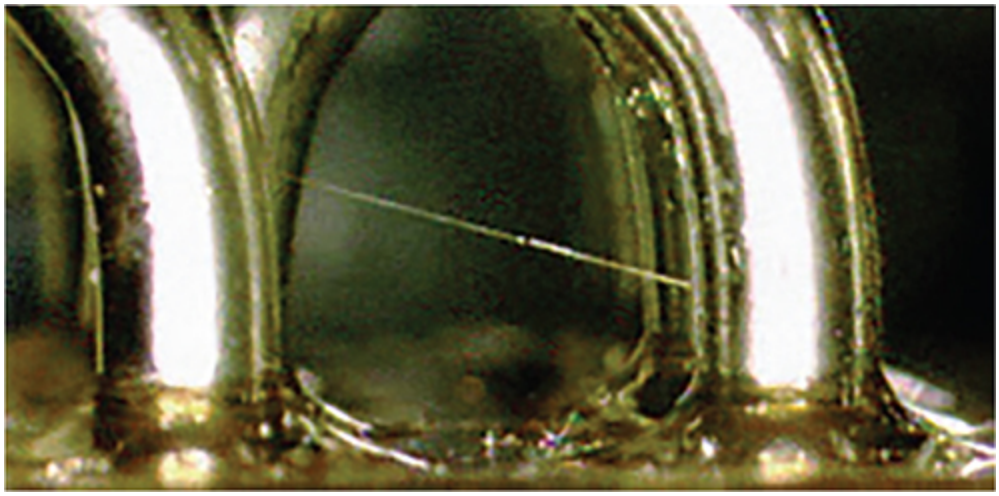


Figure 7-16: A tin whisker formation. Image courtesy of NASA Goddard Space Flight Center.

Tin whiskers can cause shorts, as shown in Figure 7-17, and can lead to very significant problems. Eleven different NASA missions have had flight computer failures because of suspected tin whisker issues. Unsurprisingly, NASA has an entire website dedicated to the study and prevention of tin whiskers (*https://nepp.nasa.gov/whisker/*).

Figure 7-17: A tin whisker shorting a connector pin to the enclosure. Image courtesy of NASA Goddard Space Flight Center.

The scary part about tin whiskers is that their formation isn't well understood, and while there are methods to mitigate their formation, there's no known way to completely prevent them. The best ways to mitigate them are to use conformal coatings and avoid pure tin or zinc. Practically speaking, avoiding tin whiskers means you should use ENIG plating when possible. (A possible exception is for RF traces, since ENIG adds extra loss at high frequencies, as discussed earlier in Chapter 6.) If using ENIG isn't possible, don't use a plating that's pure tin or zinc. Tin whiskers can form if you don't use an alloy and can cause shorts later in the life of the product.

Connectors can also cause problems with the PCB itself. Be particularly careful with 50-mil through-hole connectors on multilayer boards with a wide glass weave. The fiberglass that makes up circuit boards is available in different weights and widths. The widest commonly available width is known as 7628. Wide fiber widths are more prone to separation, and the layers can begin to peel apart internally when holes are drilled. Then when the PCB goes to get

plated, copper can seep between the layers. If these copper tendrils don't quite touch, the board will pass electrical tests at the factory and may get assembled and make it to production. But if that copper shorts out, it can cause major problems. The exact combination of a 50-mil through-hole connector and a multilayer PCB made with 7628 weave has been known to cause multiple fires due to this failure mode.

The *finish* on a PCB refers to the treatment used on the pads after etching, and it significantly affects both ease of assembly and product reliability. ENIG is just one of several finishes commonly available at most PCB fabricators. ENIG is popular because it has a flatter finish than tin/lead plating, which can reduce tombstoning and improve assembly yield. The downside of ENIG is that it's more expensive than other finishes and can suffer from a corrosion problem called *black pad*. To address this last problem, you can add palladium plating, a process called *electroless nickel, electroless palladium, immersion gold (ENEPIG)* or, sometimes, *universal finish*. ENEPIG solves the black pad problem, but it's even more expensive and sometimes doesn't solder quite as well as ENIG.

On the opposite end of the spectrum is *hot-air solder leveling (HASL)*. HASL is very common, cheap, and often the default finishing option at PCB fabricators. It can be difficult to make perfectly flat, it contains lead, and it isn't great for very fine-pitch pins. However, HASL is usually the cheapest finish you can get. Lead-free HASL is also available for designs that need to be RoHS compliant.

Organic solderability preserve (OSP) is different from the other finish options we've talked about because it's not a metal. Rather, it's an organic coating (sometimes known by the trade name ENTEK) used to keep copper pads from oxidizing after etching. OSP remains flat, doesn't contain lead, and is cheap. The downside is that it doesn't work well with plated through holes, has a short shelf life, and is sensitive to handling. Hard gold is another finish option, but you shouldn't use this unless you have a specific reason. It's expensive, doesn't go down very well, and can cause corrosion in rare cases. Hard gold is extremely durable and mostly finds use in card edge finger contacts or other contacts that experience a lot of mechanical wear over their lifetimes.

Conclusion

DFx is what separates the pros from the amateurs when it comes to developing a product. The concepts we discussed in this chapter are essential when designing something that's going to be manufactured, but many of them are also worth implementing even when prototyping or building a small number of units. The core idea of DFx is to think ahead, anticipate how your product will be built and used, and design accordingly.

8
REGULATORY REQUIREMENTS FOR ELECTROMAGNETIC COMPATIBILITY AND IMMUNITY

In the United States, one job of the Federal Communications Commission is to enforce regulations designed to keep electronic devices from interfering with each other. In this chapter we'll discuss strategies you can implement during layout to ensure your design complies with these and other regulations.

FCC regulations for consumer electronics serve two main purposes: limiting the fields that electronic products are allowed to emit and requiring that devices must remain operational even if there are interfering fields present from other devices. Together, these two requirements are commonly referred to as *electromagnetic compatibility (EMC)*, though the ability to accept interference from other devices is separately called *electromagnetic immunity*. The specific laws regarding these requirements are commonly referred to as FCC Part 15 (really Title 47 CFR Part 15). These are likely the laws that you'll need to follow, at least in the United States, but check with a lawyer or compliance lab to be sure.

Reducing Electromagnetic Emissions

If your device intentionally radiates, the FCC and other regulators around the world verify that your product meets field emission requirements by testing it in a lab. However, even if your product doesn't intentionally radiate, you're still required to meet unintentional radiation limits. Some big retailers may require that you have a third party verify that your product meets FCC requirements before stocking it. Exactly what is required depends on the country that you're selling in, so make sure to consult a compliance lab to understand what's necessary.

For your device to comply with EMC standards, you need to make sure you don't accidentally design antennas into your layout. Long traces and loops can easily radiate, for example. The fields a current-carrying loop emits are proportional to the current, the square of the loop radius, and the square of the frequency. Also, don't forget that ground and return currents are just as important as currents flowing into components.

In general, the area of the loop is what's important. A long and skinny trace that goes out but then returns directly underneath itself will have a small loop area. A trace that goes out into a large circle before coming back will have a larger loop area. The best strategy is to try to keep traces as short as you can, route them straight to their destinations (unless you need to meander for length-matching purposes), and ensure there's a return current path directly underneath or very close to the outgoing trace.

Don't put slots in your PCB. Doing so will force currents to move around the slot holes, which creates large current loops. This also holds for slots in copper planes, not just holes in the board. It can be easy to accidentally design a hole in a copper plane without realizing it. One way this commonly happens is by placing a tight row of signal or power through-hole vias. Each via will punch a small hole in the ground plane, which is fine on its own, but a tight row of these holes will merge to create a larger slot. You can avoid this problem by staggering your vias

slightly so there's enough space between them for your ground plane to pour.

Another way you may accidentally design in a ground plane slot is by routing on the ground plane layer. This gets especially tempting with a two-layer design, but it's best not to route any signals through your ground plane. Some application notes or datasheets may direct you to use a smaller, local ground plane around an IC to try to improve isolation. This is most commonly seen in ADCs. This technique generally causes more problems than it solves. In fact, there's a joke among EMC consultants that goes like this: "What do you call an engineer who splits a ground plane? A customer." You can instead accomplish what you want by moving subcircuits far apart rather than putting them on a split ground plane. This is why it's so important to think about placement first and to tightly clump subcircuits together.

The exception to splitting ground planes is when you need galvanic isolation. This is often done for safety reasons. To maintain galvanic isolation, you can't run any traces over the gap between the two ground planes, and this will prevent any EMC problems. The only things that are allowed to cross the gap are optical signals (like in an optoisolator) or fields (like in a transformer).

A common misconception is that current flows along the path of least resistance. That's only true at DC and very low frequencies. Once frequency increases, current flows along the path of least inductance. This is because, at high frequencies, impedance on your PCB is dominated by inductance. As such, a trace carrying any signal above about 1 MHz will start to have its return current flowing directly underneath it. That's why it's so important to use a ground plane under an RF trace and why a cut in the ground plane can make things worse: The return current now has to make a larger loop by moving around the edge of the cut instead of flowing right underneath the RF trace.

Bypass capacitors need a low-inductance path to power, ground, and the IC they're servicing to work properly. To accomplish this, place bypass capacitors near the IC. Smaller-value capacitors should be closest, followed by those with larger values. For high-speed applications where performance is critical, you can use via-in-pad so

there's zero distance between the capacitor terminals and their path to ground. If you don't use via-in-pad technology, then you'll still want to keep the vias as close to the pad and each other as you can. The longer the path from the capacitor terminal to the ground plane or power plane, the more stray inductance there will be and the worse your capacitor will be at bypassing.

Every single current must eventually return to its source. Remember, even if your signal's frequency is low, fast *edges* on that signal will contain lots of high-frequency content. For example, even a 10 kHz clock signal with sharp, square edges will contain high frequencies way above 10 kHz and can easily cause you to fail emissions testing if you accidentally create a big loop for the return current.

If a signal has high-frequency content (due to either fast edges or a high fundamental frequency), don't route it under components or traces that are used for input entering the PCB or for output leaving the PCB. If a signal with high-frequency content capacitively or inductively couples to an input or output signal that will leave the board, EMC problems can occur because of how long the path back to ground can be for the return current.

Stackup matters a lot in preventing EMI and EMC problems. The name of the game is containing fields, and your stackup can go a long way in preventing stray EM fields from coupling into other signals or feeding accidental antennas. It's very important to have at least one complete ground plane. Keep fast clocks, signals with fast edges, and other high-speed signals sandwiched between ground planes or power and ground planes on inner layers.

If you use more than one ground plane in your design, they need to be connected together often. The vias that are spread all around your board to do this are often called *stitching vias*. In sensitive areas, like around RF traces or between parts and signals that shouldn't couple, you can add what are called *shielding vias*, or a *via fence*. This is a group of vias that ensures a low-inductance path to ground for any return currents and prevents ground planes on multiple layers from accumulating a significant common-mode voltage. If you have any kind of high-speed, RF, or fast edges on your board, adding shielding vias can

only help you. They'll ensure that everything is at the same potential and that there's always a low-impedance path to ground for your return currents.

RF traces on the top layer using CPW or microstrip transmission lines can be really helpful when trying to debug or bring up a PCB (since you can easily add a pigtail, cut a trace, or add filtering and matching parts); however, those structures will also radiate more than a buried structure like a stripline will. All of these structures should be kept away from the edge of the PCB as fields can more easily escape into the world when high-speed signals are run along the edge of a PCB. A good rule of thumb is to leave about 400 mils between the edge of the board and the high-speed trace for any sections of the trace that run parallel to the edge.

Recall that we've been treating signals with fast rise times similarly to RF and high-speed signals because a fast rising or falling edge must have high-frequency components to it. Again, think about the Fourier decomposition of a square wave. An ideal square wave has infinitely fast edges and requires infinitely many odd harmonics to create. As you reduce the number of odd harmonics, the edges will take on a smaller and smaller slope. In electronics design, this is actually a good thing (to a point). Digital ICs will often have a timing diagram in their datasheet that will specify the minimum edge needed to function properly. You should try to reduce your clock and data line edges to be just above this minimum. The idea is that a slower edge will result in less high-frequency content in that signal, which in turn will reduce the likelihood of EMI problems. Higher-frequency signals couple into components and traces more easily than low-frequency signals because a shorter wavelength means there are more things that are a significant fraction of the wavelength and will therefore act like an antenna. The best way to reduce the speed of edges is to use either series resistors or ferrite beads. The time it takes a signal to traverse a trace should be at least five times the rise time of that signal.

Emissions are only half of the equation. Your device also needs to be immune to interference caused by other devices. In addition to the techniques discussed in this section, you can use firmware to assist.

Mechanisms like watchdog timers, memory canaries, checksums, EDAC, and status signals from peripherals can help you detect errors and respond appropriately.

Differential Pairs

Route differential signals together and length-match them. If you want them to behave like differential signals and not just two single-ended signals, their spacing and width must be precise. Use an online calculator or a tool like LineCalc from Keysight ADS to figure out the exact dimensions required. (Links can be found on this book's website.) Routing a differential pair incorrectly can lead to both signal integrity and radiated emissions problems. DDR, PCI Express, SATA, and USB are all examples of buses that have high-speed differential pairs. If you need to transition a differential pair to another layer, make sure there's a good current return path near the transition. Figure 8-1 shows the right way to do this.

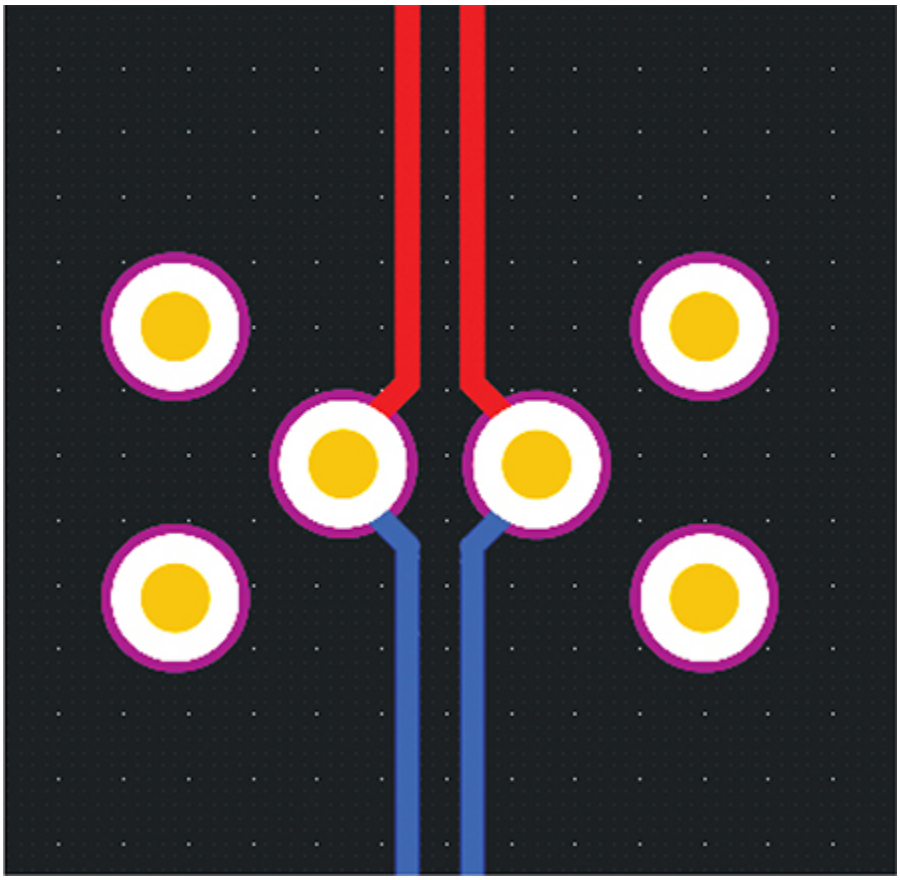


Figure 8-1: A differential pair via transition, with ground vias near the signal vias to provide a low-impedance path for return currents

If you're routing very high-speed signals (greater than about 5 gigatransfers per second) and your trace runs in a straight line for more than 4 inches (10.16 cm), you need to be concerned about the *fiber weave effect*. This is caused by one trace in a differential pair running directly on top of the fiber-glass weave in your PCB substrate and the other trace running directly on top of the space between weaves. This results in two different values of relative permittivity and can cause signal integrity problems. Figure 8-2 shows what a route vulnerable to the fiber weave effect looks like.

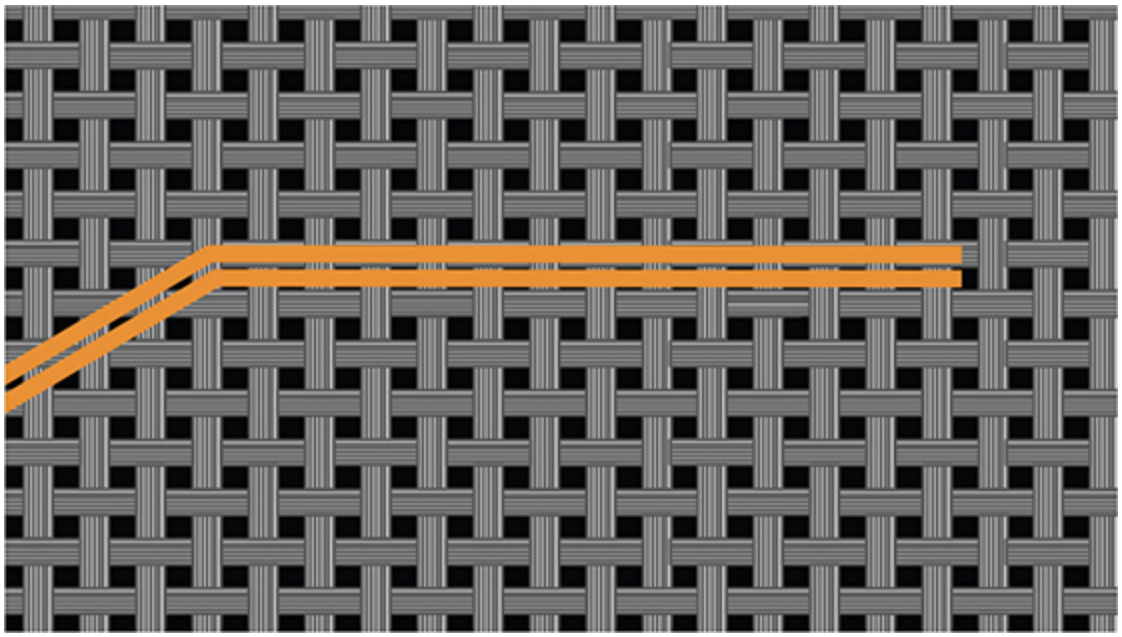


Figure 8-2: The fiber weave effect is caused by different relative permittivity under each trace.

To combat this effect, use a zigzag pattern like in Figure 8-3.

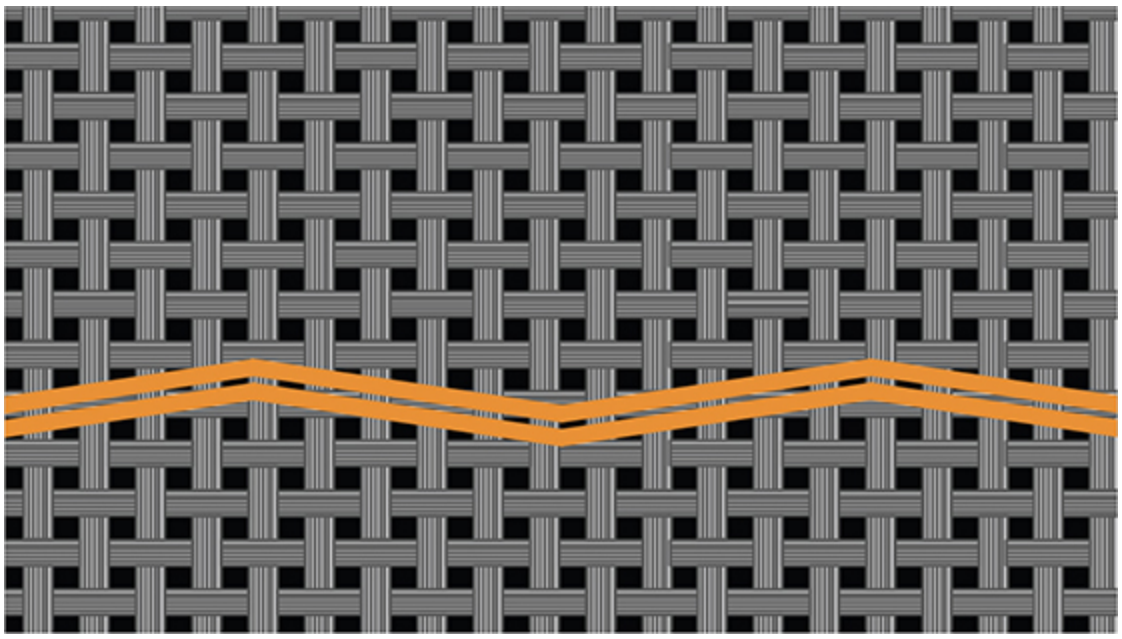


Figure 8-3: A zigzag routing method used to combat the fiber weave effect

The zigzag pattern ensures that both traces experience roughly the same relative permittivity throughout their paths.

Signal Integrity

Physical isolation can be helpful in preventing noise and EMI from one component or subsystem from contaminating other components or subsystems. Switching power supplies can be large sources of noise due to the fast switching of the FET. Keep switching power supplies isolated from one another and far away from antennas, RF signals, and any sensitive analog components or high-speed digital components.

Another important way to keep your signals clean is to place filters for incoming and outgoing signals as close to the edge of the board as possible. If a "dirty" input signal makes it to the middle of the PCB before it gets filtered, that noise can couple into other signals and components before it's filtered out. Similarly, if an output signal is filtered near the middle of the board instead of the edge, noise in the bandwidth of the filter can couple back into the signal from other components and signals on the PCB before it leaves. In both cases, you're defeating the purpose of the filters. This also applies to cables, not just signals on a PCB. If a cable with a filtered signal is run next to the cable with the unfiltered signal, the filter won't do much good.

Other sensitive signals need to be physically isolated, too. Differential pairs and traces that are length- or impedance-controlled shouldn't have any other signals running parallel to them for long distances. This can result in an accidental coupling effect or can throw off the otherwise carefully controlled structure of your traces.

If a trace is impedance-controlled (using a structure like stripline, micro-strip, or CPW), you must maintain a certain separation distance on all sides of the traces to achieve the expected performance. There are lots of online calculators you can use (see the book's website for links), or you can use a tool like LineCalc, which comes with Keysight ADS. Whatever you use, these calculators will specify a separation distance between every feature. If you aren't using a particular structure but are still concerned about accidental coupling, use a separation distance of three to five times the width of your trace. After that distance, the field strength is low enough that coupling usually isn't a problem.

You should try to keep all traces as short as you can, but high-speed or fast-edged signals are especially important to keep short. The longer those traces are, the more likely it is that they'll couple into other traces.

Connectors

Connector placement matters. Connectors present opportunities for energy to radiate out into the world because most connectors and cables happen to act like good antennas for at least one frequency. One common way this can happen is if you have connectors on opposite ends of a PCB. Those two connectors can easily act as a dipole antenna and cause EMI problems. To avoid this, you should place all connectors on the same side of the PCB. Sometimes this is impossible for product design reasons, but it will significantly mitigate the risk of EMI problems.

If you have multiple signals going to a single offboard device, run them all through the same connector and cable. Splitting up groups of signals up and running them through two connectors and cables before recombining them has a similar effect to putting connectors on opposite sides of the PCB. It can create common-mode voltages that turn the cables into antennas. Other techniques that can help reduce cable emission problems include common-mode filters, ferrite chokes, and shielded cables.

It's also important to keep any signals that are high-speed, are RF, or have fast edges away from other connectors. High-frequency signals generate a lot of harmonics and can more easily induce a voltage in connectors and cables, causing them to act like antennas. If you need a high-speed or RF trace to leave the PCB, use a controlled structure like stripline, micro-strip, or CPW and keep source or generating circuitry away from the connector. Controlled-impedance structures will give you more contained and predictable EM fields. This will make it easier to shield and mitigate any EMI problems that result.

A properly designed cable should have a ground wire immediately adjacent to every signal wire and every wire carrying power. If you don't do this, you'll be creating a large current loop and will likely have EMI

problems. For even better performance, twist each signal or power wire with its adjacent ground wire.

Power Planes

When routing power, utilize a power plane. That way, instead of routing traces all over the place (inviting the opportunity for coupling and radiation of high-frequency noise into other traces and components), you can just drop a via straight down to your power plane. If you have more than one voltage in your design, use multiple planes on the same layer. Circuits using the same voltage should be clumped together anyway, and keeping all power planes on the same layer prevents power planes from overlapping and capacitively coupling noise into one another. Keep a healthy clearance of 100 mils between plane edges. This helps minimize coupling along the edges of the planes. However, if your application has high voltage, you may need a larger clearance to prevent arcing.

Another reason to use a power and ground plane is that simplifying your routing can also improve performance. Rather than connecting a pad to ground through a long trace, having a power and ground plane allows you to remove the extra inductance of that trace by simply dropping a via right at the pad and only incurring the stray inductance of a via. This can be really helpful for the performance of power supplies, for example. Bypass capacitors are also more effective over a larger frequency range when using this technique.

It's important to understand that the energy of any electrical signal is contained in its fields. That means that if you have a power plane above a ground plane, the energy will be stored in the fields within the dielectric between them. Once you think about it this way, it makes sense why it would be a bad idea to insert another signal layer between your power plane and ground plane. The signal layer would run right through the fields of your power plane and the two will crosstalk.

Shielding

If you're particularly worried about a certain area on the PCB radiating, or if you have some very sensitive circuitry, you can use a can-type shield. *Can shields* are little metal lids that sit on top of the PCB and keep electric and magnetic fields from getting in or out. You can either solder the shield perimeter down directly to the board (a single-part can) or solder down clips that the shield presses into (a two-part can). The advantage of using clips is that it's much easier to rework, debug, and measure the circuit being shielded. Inspection with a single-part can in place is very difficult. Using clips is slightly more expensive because you need to buy both the clips and the shield, but it's still highly recommended. Even if you have the design finalized and totally perfect, using shields that are easily removed will make repairs easier when units fail in the field and are returned.

Shields can either be solid or have holes in them. As long as the holes are very small compared to the wavelength of the signals you're trying to block, the shield will still work fine and you get the advantage of a little more air circulation for heat dissipation. For the shield to be effective at a particular frequency, the size of the holes must be no more than one-tenth of the wavelength of the highest-frequency energy you're trying to keep in or out of the shielded area. Similarly, the spacing of the clips on a two-part can is very important since you can accidentally create slot antennas that cause EMI problems.

Shields also come in shapes other than squares or rectangles. Most shield manufacturers can make custom shields of whatever dimensions or shapes you need. However, if you're on a budget, look for premade shields and design your sensitive circuits to fit within those areas. On RF circuits, it's common to need a can shield with multiple compartments to shield different parts of the RF circuit from each other. This will require a custom shield.

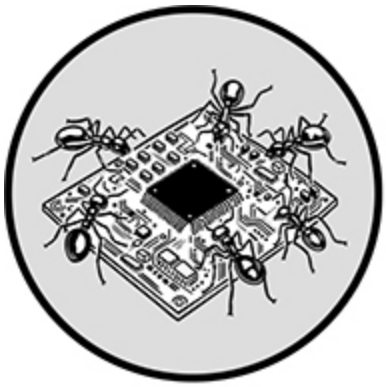
If you have a shield around your entire assembly, the only correct place to electrically connect it (including the shield of a shielded cable) is to the metal chassis of the device and never to the internal ground plane of the PCB inside. If the chassis is already connected to your PCB ground plane, that's okay because your shield will end up being connected to the ground plane, too. If you're using a PCB can shield, it

will always be connected to the PCB's ground plane. Remember that you're trying to build a Faraday cage. You want your shield to be a continuation of the metal enclosure to shield the electronics inside. Shielding becomes much more difficult if your enclosure is plastic instead of metal. In that case, the next best thing is to have an internal metal chassis that does the shielding and has a plastic case around it. It is surprisingly easy to build something that looks like it should be a good shield but actually isn't.

Conclusion

There are many companies that exist just to solve the EMI and EMC problems that designers encounter, and those companies make a lot of money. You can save yourself a lot of time, stress, and money by designing your product to pass regulatory testing on the first try. Take the time to consider what effect your design choices will have on EMI and EMC before you fabricate your boards. Building a solid understanding of the physics going on within your product, beyond the simplified mental models most of us operate on, will make it much easier to succeed.

9
COST ENGINEERING



Cost engineering is the design phase where you try to reduce not only the cost of components but also assembly and repair costs. In this chapter, we'll look at cost-engineering strategies. We'll discuss ways to pay less for components and find lower-cost replacement components, and we'll look beyond shaving off component costs to find other ways of reducing your cost of goods sold (COGS).

In principle, the benefit of cost engineering is simple: The better you are at it, the more money you stand to make from your product. Good cost engineering can even be what separates successful products from failures. Many products have tanked despite having great technology because they were too expensive for the market.

Overall Strategy

Don't start cost engineering too soon. Design to meet your product specifications first, and only then should you begin to look for replacement parts that still meet all specifications. That doesn't mean you should be totally removed from the concept of component cost during the prototyping phase, however. While the prototyping phase is too early to try finding something like cheaper connectors, you also

shouldn't fall into the trap of designing an architecture around expensive, irreplaceable components that will require a complete redesign during cost engineering. See Chapter 2 for more tips on choosing the right components.

Cost engineering has failed if it causes any of the specifications to no longer be met. Cost engineering has also failed if it significantly degrades the user experience, transforming a long-lasting product into cheap crap. Two products may have identical functionality, but if one is hefty and has a brushed metal enclosure and the other is lightweight and has a plastic enclosure, the user experience will be different. One will connote a feeling of luxury and importance, while the other will come across as inferior and cheap. Think about the impression you want to make on your users. There isn't one right answer here: It might be okay if something feels cheap if it's designed to be disposable, for example.

Cost engineering doesn't happen in a vacuum. Approach it as a collaborative effort, with your mechanical engineers and industrial designers making major contributions. Talk to them to see how your design is constraining their work, and see if there are any changes you can implement that can reduce cost for another member of the engineering team.

Overall, the best way to reduce the cost of your design is to remove things. The most obvious example of this is to take entire components out of the design, but this isn't always possible. Instead, think about the other things you can reduce: the number of bits in ADCs or DACs, the amount of memory in your EEPROM, the resolution of your display, and so on. Cost engineering can require just as much creativity as the design itself.

Cost engineering by looking for replacement ICs can get dangerous. For something like a power supply IC, the process of qualifying and testing the new part should be fairly straightforward. By contrast, for something like an EEPROM or other peripheral, new firmware will probably need to be written, which will significantly impact your schedule. If you decide to switch microcontrollers or processors, a huge amount of work will need to be done to integrate the new part. Any cost

engineering of this kind should be done as close to the beginning of development as possible. In this case, the importance of a low BOM cost should be expressed in the requirements so that only one set of firmware is ever written.

How you approach component cost engineering depends on your production size and your market. For a very large production volume, saving a penny on one component can result in a lot of money saved in total. However, for a small production volume, many cost-saving strategies aren't worth the time needed to source a new part. Before you go looking for a capacitor that costs $0.005 instead of $0.01, calculate what you'll actually be saving.

As with everything, the more you buy, the cheaper it is. You can see some of the price breaks for volume on Mouser or DigiKey, but they usually only go up to about 10,000 pieces. Past that, you should talk to the manufacturer or distributor sales reps. They'll usually be able to beat the prices at DigiKey and Mouser at both large and small quantities, so it's worth it to contact them and get quotes.

Cost Reduction in the Schematic

Your schematic contains all of the electrical components you need to buy to create your product, so it's an obvious first target to reduce the product's cost. Let's talk about ways you can edit your schematic to reduce the BOM cost while still achieving your product requirements.

When possible, try to select common parts that have many available substitutes. Sometimes you can find parts with similar or identical functionalities made by different manufacturers that are pin-compatible. Even parts that are similar but not pin-compatible can sometimes be made to work on the same PCB through careful use of jumpers and 0 Ω resistors. This can allow you to reduce your BOM cost by giving you more options and leverage for price negotiation. It also makes your design more resilient to supply chain problems.

Don't be afraid to put down footprints for parts even if you're not sure you'll use them. You can always just leave them unpopulated. If you need a DC connection through the unpopulated footprint (if you

remove a series capacitor, for example), you can simply populate a 0 Ω resistor of the same size. Maintaining a DC connection through an unpopulated IC footprint is trickier, requiring you to design in that option. Having the flexibility to try different circuits or add and remove parts can be worth it, though, allowing you to experiment while keeping you from having to fabricate another PCB iteration.

You can use 0 Ω resistors to jump over traces on the top layer, effectively giving you another layer on the PCB almost for free. There are also purpose-made jumpers for doing this. Search for "Surface-Mount Printed Circuit Jumpers" by Keystone Electronics. For RF signals, there are purpose-made RF jumpers that can be used to hop over perpendicular transmission lines without significant loss or reflection. Kyocera makes the MLO RF-DC SMT Crossover, which is a surface-mount component designed to allow a critical transmission line to cross a DC trace. Kyocera also makes the MLO RF-RF SMT Crossover, which allows two RF traces to cross. Both of those crossover components are good up to 6 GHz.

While it may not always be useful to try to save fractions of a cent on components in a small production run, both large and small production runs can benefit from reducing the required number of *unique* parts (sometimes called the number of *line items*). For example, say you have a 10 kΩ pull-up resistor on one GPIO pin and a 15 kΩ pull-up resistor on a different GPIO pin. You probably don't really need two separate values. Standardizing to a 10 kΩ resistor on each pin will reduce the number of feeders used on the pick-and-place machine during assembly. Manually assembling the boards will be faster, too. In both cases, a lower unique part count will result in lower assembly costs.

The exception to this rule is high-tolerance parts, which are expensive and should only be used when needed. Don't accidentally change out a part that needs to be high-tolerance with a standard-tolerance part in an attempt to reduce unique part count. Similarly, don't change all values of a single part to be high-tolerance if only one of them actually needs to be high-tolerance.

Even if you've standardized on a particular value, discrete components of the same value will often have different costs depending

on the package they come in, so you need to consider that factor as well. Usually 0603-size discretes are a good trade-off of cost versus ease of assembly and rework. You can also check to see if any parts you're using are cheaper in a different package.

The cost of connectors can add up quickly, so use as few connectors in your design as possible. In "Connectors and Cables" on page 44, we discussed a few ways to do this. One was to use soldered wires and the *hot-bar* soldering technique, where a hot bar (sometimes called a *thermode*) is laid across the wires on the pads and all the solder melts at the same time. It's fast, suitable for mass manufacturing, and can be used on wires or flex cables. Another option we discussed was using a card-style connection, where the PCB has exposed contacts along one edge that plug into a receptacle on another board. This reduces the parts requirement from two connectors and a cable to a single connector. However, connections that require a high signal integrity (such as high-speed or RF signals) will have to use a properly rated cable and connector.

One great place to save money on connectors is the programming connector. You'll use the programming port all the time during prototyping and development, but probably only once to flash the firmware during production assembly. Rather than paying for a connector that gets used once, you can use a component-less programming header. A product called Tag-Connect is a popular example. A special cable with pogo pins and plastic clips latches onto a specially designed footprint on the PCB, makes good contact, programs your board, and is then removed. This has the advantage of not adding the cost of a connector to your BOM and makes it easy to access the programming pins again later during repair or maintenance. You can also integrate a TagConnect programmer (or just some pogo pins that accomplish the same thing) into your bed-of-nails test jig and use it to both program and verify functionality of your product during production. See Chapter 13 for more information on how to do that.

Here are some other component-specific factors to consider to help cost-engineer your schematics:

Arrays One way to lower the total part count is to use resistor and capacitor arrays. These will work only if you have several resistors or capacitors of the same value that are physically close to each other. The advantage is that you save assembly cost because it's faster to populate a single part than to populate several. Keep in mind, though, that using an array may increase your *unique* part count by one if not all resistors of that value can use an array. For example, if you have five 10 kΩ resistors on a PCB and four of them are close enough to use an array, you've now increased your unique part count by one, since you need a single 10 kΩ resistor and a 10 kΩ resistor array. You could prevent this by using arrays even in places where you don't need every resistor in the array. Resistor arrays cost more than individual resistors, though, so you'll need to do a little cost analysis to determine whether it's worth it to use an array at all.

Load switch ICs You may be tempted to replace a load switch IC with a FET. Given that load switches are designed to have a very low on-resistance and typically have ESD protection built in already, it's usually not worth it from a cost perspective to replace a load switch IC with a FET. Load switches are already very inexpensive and usually cost less than buying a separate FET and ESD diode.

Oscillators A lot of modern microcontrollers and processors have an internal oscillator. If your application can accept the reduced accuracy and lower clock rate of an internal oscillator, you can eliminate the need for an external crystal oscillator. Many microcontrollers also have GPIOs that can act as programmable clock outputs. This allows you to distribute a clock to other devices, potentially saving you from buying additional oscillators.

ESD diodes If you need a higher level of ESD protection, stacking two ESD diodes in series is sometimes cheaper than using a single ESD diode with a higher rating.

These approaches can help get your BOM cost lower than if you were to just try to find the best bargain on each component. Rethinking what components should even be in the schematic ought to be the first

step, followed by hunting for suppliers that will give you the best price on them. Reducing your component costs will allow you to pay less for your design even at low volumes.

Cost Reduction in the Layout

The easier a PCB is to manufacture, the cheaper it will be. This sounds obvious, but what makes a PCB easy to manufacture? The simplest answer is to use standardized design choices that the PCB manufacturer stipulates, like a minimum trace width, minimum space between copper, and minimum via size. A common set of requirements for this cheapest tier of manufacturing might be a 6-mil minimum trace width, a 6-mil minimum space between copper, and a 10-mil minimum via size. These requirements vary by manufacturer, so look them up on the website of whichever you decide to use. This is an area where making a slight change, like widening a trace by 1 mil, might not make any difference to your product's performance but can impact your manufacturing cost greatly.

Once you start adding features like HDI, blind vias, buried vias, micro-vias, exceptionally tiny trace widths, thicker copper pours, and plugged vias, the costs will quickly start to mount. It usually doesn't matter how *many* of one of these features you add—the cost increase will be the same regardless. For example, as discussed in Chapter 6, adding a single blind via won't be any less expensive than adding 100 blind vias (assuming they're all between the same layers). The cost increase comes when you go from 0 blind vias to any number greater than 0.

Following manufacturer-recommended design rules can also speed up manufacturing time. A standard design that uses tolerances and techniques the manufacturer can quickly execute can usually be fabricated in a couple of days, or even as quickly as a single day in some cases. By contrast, more complex designs can take weeks.

Another way to keep the cost of your PCB down is to make sure it's optimized for efficient panelization. We talked about this in Chapter 7. Fitting as many copies of your board as possible onto an 18×24-inch

panel will minimize the amount of material that needs to be scrapped. For example, a 5×7 PCB will be much less expensive per square inch than a 5.8×6 PCB. They both have a surface area of 35 square inches, but the 5×7 board can comfortably fit nine copies on an 18×24 production panel, while you can fit only six of the 5.8×6 boards in the same space. That means the 5.8×6 board will cost 50 percent more than the 5×7 board. The size of your board, and even its shape, will have a large effect on how much your PCB will cost.

Remember, the fabricator uses the outer 1 inch of the production panel for tooling and copper thieving. That leaves a remaining usable space of 16×22. It's possible to use a production panel that has dimensions other than 18×24, but this costs extra money. The other advantage of using an 18×24 panel is that every PCB fabricator on the planet can build that size, so it's much easier to build your design with different fabricators.

NOTE

If you're just building prototypes or a few copies that you won't produce again, don't bother spending the time to optimize your panel.

Your PCB layer count also impacts cost. A four-layer PCB is typically about double the cost of a two-layer PCB, a six-layer board is about 50 percent more expensive than a four-layer board, and an eight-layer board is about twice the cost of a four-layer board. Keep your layers to the minimum that you need. That said, if you try to reduce your layer count to below what you actually need, you'll end up spending more to mitigate EMI problems than you would have spent just adding another couple of layers. You need a ground plane layer next to every signal and power plane layer, unless you have enough experience to know when you can break this rule.

In general, you should try to design PCBs with all the parts on a single side. This will save you time and money in assembly, since only one side needs to be reflowed, and it also makes building test fixtures much easier. Even placing a small number of parts on the bottom side of

a board would require the boards to go through a whole second reflow. Similarly, try to use only through-hole or only surface-mount parts. This will save your assembler time since they'll only need to run through a single process. Mixing surface-mount and through-hole parts on a single board is certainly possible, but be aware that it'll be more expensive.

If you must place components on two sides of the board, try to use only surface-mount components. If you use a mix of surface-mount and through-hole, your board either will need to be both reflowed and wave soldered or will require some manual assembly. Wave soldering is sometimes done at a higher temperature than reflow soldering, so make sure your surface-mount parts are rated for that higher temperature. Also keep your surface-mount components farther than 250 mils from your through-hole component leads if possible. Any closer and your board will need either to be manually assembled or to use selective wave soldering, both of which add cost.

The PCB substrate you use can also have a big impact on cost. This is most relevant to RF designs, where the PCB material must have a particular relative permittivity and loss tangent. One option for reducing substrate cost in a design is to use modules. Instead of manufacturing the entire PCB on the expensive substrate, use it only for the part of the circuit that needs it and treat that part as a module that plugs into the rest of the electronics. You can accomplish this by fabricating a separate, smaller board on the expensive substrate and using a connector to send signals back to your main board. Or, if you only need to route low-speed signals to and from your smaller board, make it a castellated module that's soldered on top of your main board. This may not end up being cheaper for small designs, but for large designs it can ensure you're paying for expensive substrate only in the places where you absolutely need it.

If you need only a subset of your layers to be on a particular substrate, you can design a stackup that uses more than one core material type. Figure 6-15 previously showed an example board that uses Rogers 4350B on the top two RF layers and a standard FR-4 type material for the rest of the layers. Depending on how many boards

you're making and what substrate you're using, this can be cheaper than making the entire board from a more exotic material.

Another trick to save money on PCB costs is to use a single PCB design for multiple product versions. This works only when multiple versions are mostly identical except for a few auxiliary features, but it can be very powerful. Let's say you have two versions of a product, one with wireless connectivity and one without. Rather than design and fabricate two PCBs, design the more complete version (the one with a wireless IC), and then simply mark those parts as DNP (do not populate) on the BOM for the simpler version. Some chips with different functionality have identical footprints and pinouts, which you can also leverage in this way. If you want to sell a version of your product at a lower price with less memory, you can often find memory in the same family with the same pinout and footprint. All that's needed is to change one feeder on the assembly line and to flash a different firmware version. You can even have your firmware check for the presence or absence of a pull-down resistor on a spare GPIO pin to automatically detect the product version so you don't need to reflash firmware for each version.

Cost Reduction in Assembly

Not every distributor sells the same parts for the same price. There are several different services you can use to compare component costs from different distributors. Personally, I like to use Octopart.com. This site allows you to upload your BOM, and it will automatically search through a list of dozens of distributors to see who has the part in stock, if they have enough for the number of boards you want to build, and what the cost per part is at that quantity. This is a great way to predict what the manufactured cost of your design will be at production volumes, even before you actually want to make that many. Links for Octopart and other similar sites are available on this book's website.

Another tool you can use to predict component, PCB fabrication, and assembly production costs at volume without waiting days for a quote is CircuitHub.com. This turnkey PCB manufacturer lets you

upload your design files and get an instant quote for anywhere from a single board to 10,000 copies. For example, Figure 9-1 shows two CircuitHub quotes for the same device: one to manufacture a single board and one to manufacture 780 copies of that same board.

Even if you don't end up ordering from CircuitHub, it's quite instructive to upload your design and play around with the sliders to understand how quantity affects cost in your design.

A BOM contains the list of parts that are used to manufacture your device, but the BOM is only a subset of the COGS. One aspect of the COGS that's directly affected by a device's electrical design is any manual assembly that needs to be done. The biggest item here will be hand soldering. Other manual steps may include applying glue or adhesive, screwing in fasteners, or building and installing wire harnesses. The fewer steps, the cheaper the assembly will be.

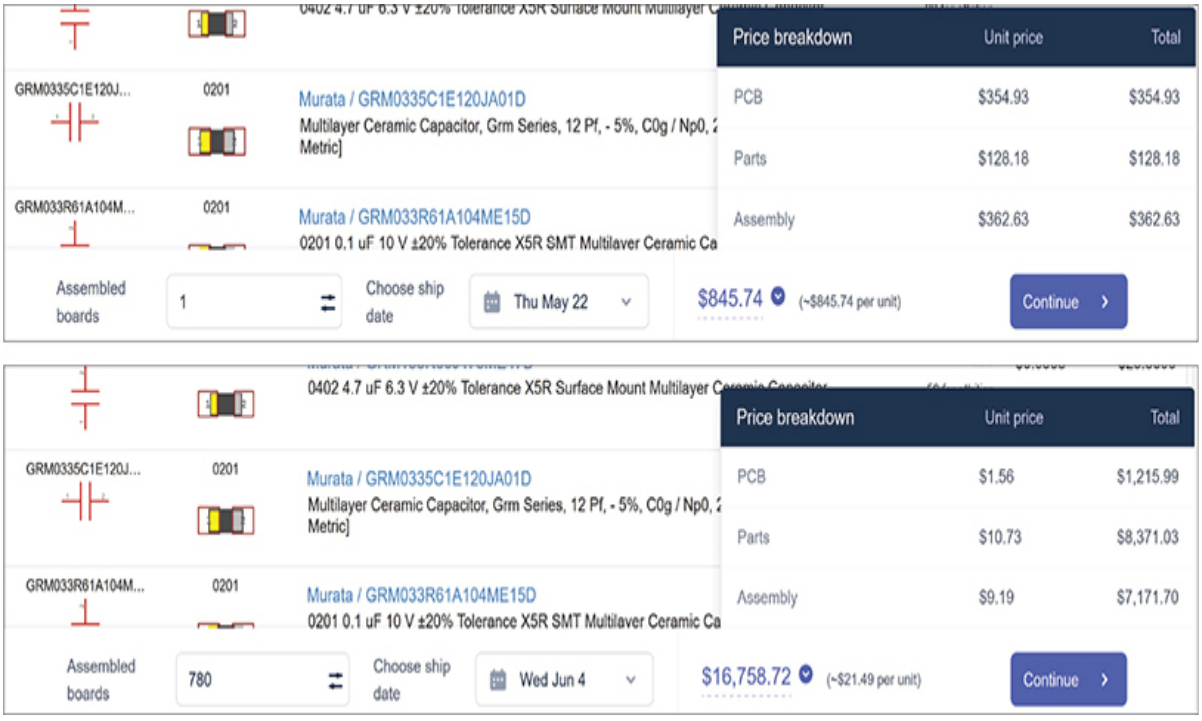


Figure 9-1: CircuitHub quotes for manufacturing a single board (top) versus manufacturing 780 boards (bottom)

There are also strategies for reducing the cost of the enclosure itself, but those are outside the scope of this book since your mechanical engineer will likely be handling that. However, it's important that you

work with your mechanical engineer on this task because changes to an enclosure to reduce manufacturing cost will very likely impact your design, perhaps in ways that your mechanical engineer won't anticipate. For example, changing the enclosure material can affect the device's shielding properties.

Use as many industry-standard measurements and dimensions as you can. If you design a PCB panel to be 1 cm bigger than the rail size at the factory, the manufacturer is going to charge you a lot more because they have to retool their line for you. If you pick a solder mask that's a different color than the one they already have in their machine, they're going to charge you extra. Talk to your factory and ask them how you can reduce their cost of labor and materials.

A method called *chip-on-board* can help reduce the cost of the ICs themselves. Normally, you buy a chip in a package from the manufacturer. The manufacturer puts their die in a lead frame that contains the external pins and then molds a black plastic epoxy shell around it, resulting in the familiar look of a chip. However, you can sometimes cut a better deal with chip manufacturers by buying just the die by itself and epoxying it directly to the PCB—hence chip-on-board. Next, you need to use an automatic wire bonding machine to place tiny wire bonds between the pads on the die and the PCB. Then you cover the entire thing with a glob of protective epoxy. If you've ever taken apart a calculator and noticed the black blob in the middle of the PCB, you've seen a chip-on-board in the wild. This process makes financial sense only if you're doing a very large production volume, but in those cases, it can provide a significant reduction in BOM cost.

Conclusion

Once you have an accurate estimate of the number of devices you'll be making, you can determine the best way to reduce the cost. Design simplification is the best way to reduce your COGS, as long as you don't cause the product to miss requirements or significantly alter the user experience. Pay only for the component performance you need, and avoid over-designing. Try to use the same part for multiple purposes.

And remember, the design needs to work before you start to cost-engineer it.

PART III
BUILDING

Chapters 10 through 14 teach you how to build your product in a repeatable way. You'll learn about how the fabrication, assembly, and test processes work. We'll also talk about how to build an electronics lab, tips for rapid prototyping, and how to troubleshoot electronics. Finally, we'll cover the practical, real-world lessons that are almost never taught in engineering school.

10
PROTOTYPING

You can't make it better until you make it work.
—Akin's Laws of Spacecraft Design

Effective development is driven by risk reduction, and one of the best ways to do that is by prototyping your design. In this chapter, we'll discuss how to decide what to prototype and how to implement that prototype quickly. We'll talk about the planning and design phases, as well as how to construct your prototype in the minimum number of days. Above all, the goal is to ensure that any prototype you make is a meaningful and useful tool for investigating an idea.

Prototyping allows you to experiment with different solutions and pick the best one—not by guessing but through results. Good prototyping skills also help increase your development speed, so prototyping is a valuable practice to cultivate if you want to bring a successful product to market as quickly as possible.

The Prototyping Mindset

Prototyping is all about being "close enough" and approximating more complete, fleshed-out systems. You need to use your judgment to decide where the line is with respect to development speed and accuracy of

results. Prototyping also requires a MacGyver-like intuition and inventiveness to use the materials that you have to prove out the core of an idea. A good prototype may look nothing like the final version of a product, but it will still prove or disprove your hypothesis.

Every prototype should have a purpose. Ask yourself what questions you're trying to answer with each prototype, and stay focused on answering those questions. This applies to each individual prototype, but it isn't to say that the prototyping process as a whole should be so single-minded. On the contrary, you'll have the most success prototyping when you learn to keep an open mind and think about the greater context around what you're building.

During the prototyping stage, it's very tempting to get pigeon-holed into one design or approach, but this is the opposite of what you want. You should use the prototyping process to explore different approaches that take separate technological paths. Try different parts and techniques wherever possible. Do experiments. Spin cheap PCBs to test ideas. Measure things. Take the initiative to do small tests and explore new products, even if they're half-baked.

If you're having an unusually difficult time getting one approach to work, eventually you'll need to put it aside, at least for a while. Sometimes you'll need to abandon that approach altogether. A great example of this is the early development history of SpaceX's Starship vehicle. In 2018, Elon Musk made the decision to switch the construction of the Starship from carbon fiber to stainless steel. This was after announcing that it would be carbon fiber and holding a huge press event to discuss the design. SpaceX had also already bought and received a massive carbon fiber tooling part and was already building and testing huge carbon fiber pressure vessels.

Despite publicly committing to a design and spending tens of millions of dollars going down that path, Musk made the call to scrap all of it and start over with a totally different approach. This was because the carbon fiber manufacturing process was taking too long. Musk spoke about this a year later, in an interview with Tim Dodd on September 28, 2019, at a SpaceX Starship Update event:

> Musk: If a design is taking too long, the design is wrong. And therefore, the design must be modified to accelerate progress. One of the biggest mistakes made in advanced development is to stick to a design even when it is very complicated, and not strive to delete parts and processes. . . .
> Dodd: So you're definitely not a sunk cost fallicist [*sic*]?
> Musk: No, definitely not.
> Dodd: You're like, "This is clearly the new path forward, let's hop on it."
> Musk: Yeah. Is it in the future or not? If it's not in the future, who cares?

Starting over when an approach isn't working is important, but you can avoid such dramatic restarts by parallelizing the development process. Instead of having the entire team follow one path, and then another path if the first one doesn't work, have multiple people work on multiple approaches at the same time. Bring up a low-fidelity, proof-of-concept prototype for each approach before moving to the next revision of that approach or to a new approach. For expensive, hard-tech development, these low-fidelity prototypes might just be simulations. The point is to de-risk the path to fulfilling a requirement. Breadth-first search is better than depth-first search.

In the same interview with Dodd, Musk talked about another key part of the prototyping mindset: taking a big-picture view of the goals of a project and not being afraid to question assumptions. Musk said:

> Everyone needs to broadly understand how the entire system works so you don't get subsystem optimization. The product errors reflect the organizational errors. . . . One department will design for the constraints given to them by another department without calling into question those constraints. . . . You should actually take the approach that the constraints given to you are guaranteed in some degree to be wrong. . . . You know they're not perfect. . . . Question your constraints. It does not matter if the person handing you those constraints won a Nobel Prize. Even Einstein was wrong some of the time. Question your constraints. It's extremely important.

This concept that product errors reflect organizational errors is known as Conway's Law. Originally stated, it says, "Organizations which design systems . . . are constrained to produce designs which are copies of the communication structures of these organizations." As an example, remote teams tend to produce code that's more modular than teams where everyone sits in the same room. Musk suggests that the "seams" of an organization, the interfaces where different departments meet, can easily cause the organization to turn into a simple machine that almost

blindly accepts and executes what each of the other departments says rather than thinking holistically and questioning *why* things are done a certain way and what drives the apparent constraints the department has declared. Musk again:

> One of the biggest traps for smart engineers is optimizing a thing that shouldn't exist. When you're going through college, you have to answer the question the professor gives you. You don't get to say "This is the wrong question."

Progress counts only if you're moving in the right direction. It's critical that you understand *what* to build. Engineers may think that the goal of prototyping is to reduce only *technical risk*, the risk that a product will fail because it doesn't function as expected, but you also need to be incorporating as much product feedback as possible into your prototypes to reduce the *product risk*, the risk that the product will fail because nobody actually wants it. Always remember that hardware serves the user. For each iteration, ask yourself: What changes am I making that are a direct result of talking to users? There had better be something!

At larger companies, the software, hardware, and product teams are all interdependent, working in tandem to deliver something that users actually want. The product team acts as a proxy for your users. They inform software and hardware decisions based on their research and what they learn by talking to users. The hardware team informs product and software about what capabilities are possible and what's realistically feasible (or more often, not feasible). Likewise, the software team informs product and hardware about possible new features and what planned features are feasible.

The design process is a conversation between the product and engineering teams. For example, say the engineering team pushes back on a requirement that the product team has heard from users. The product team might go back to users and determine whether that requirement can be relaxed or replaced, which allows the engineering team to reduce the cost of the product or get it to market faster. You can't create a successful product with a siloed product team that throws requirements "over the wall" to engineering. You also can't create a successful product with a stubborn engineering team that can't

negotiate with the product team. When everyone is working together correctly during the prototyping process, the end product has the most valuable features possible—features that users actually want.

The Minimum Viable Product

A *minimum viable product (MVP)* is a prototype that has the bare minimum of functionality and features needed to be something that your users want. Once you've determined the purpose of your device, based on your understanding of what your user wants or needs, and identified the high-level constraints your design will have, you should decide what your MVP would look like. Then build it!

Your MVP doesn't need to be highly robust, but it does need to work well enough for you to be able to test the core of your idea with a small group of users. When I advise companies, I always tell them that building something now and getting it out the door is better than waiting to build something perfect (unless you're talking about a safety-critical or medical device). Just build it! In the process, you're allowed (and even encouraged) to cheat however you can if it helps you try out your ideas quickly. Make a breadboard prototype, throw an Arduino or Raspberry Pi and some sensors inside a 3D-printed enclosure, or use parts from electronics hobby sites like Spark-Fun and Adafruit. Your MVP is probably going to be a bulky tangle of wires, look unprofessional, and may contain duct tape as a structurally significant part of the design. That's all okay, especially if it lets you quickly go from an idea to a working prototype. The important thing is to have something you can test.

Although you should take shortcuts where you can, you also need to take the time to make your prototype safe. A quick, hacky prototype is no excuse for putting a device that could hurt someone into the hands of your first users. This will mean different things for different products, but you need to write your final specifications with safety in mind.

If you're trying to eventually sell what you're building, first try to sell your MVPs. You'll have to make them by hand, and you'll likely feel slightly ashamed about charging people money for them, but selling

your prototypes is the best way to see how dire the problem is that your users are facing. As I mentioned in Chapter 1, if the problem is bad, users will take even a shabby version of a good solution over whatever they're currently doing to deal with the problem. The other thing that charging money will do is guarantee honest feedback from your first users. Since they have skin in the game, they won't be afraid to tell you if something about your product is bothering them. However, do your best to give them realistic expectations: They will encounter problems and bugs. The way to maintain a good reputation is to be super honest and transparent, acknowledge the problems, and address them as quickly as you can. You can actually build a lot of loyalty with your early users when they see how much you care about getting the product right, how fast you're moving, and that you're actually listening to what they're saying.

If you need to make a lot of MVPs (on the order of hundreds), or if you need access to a special process or material that's difficult to get, try finding an existing product that does most of what you want and modify it for your special case. You can do this by either buying and hacking products already on the shelf or working directly with the manufacturer. I've done this by searching Alibaba.com or Aliexpress.com for products close to what I wanted to make, or products containing major parts of what I wanted to make, and then reaching out to several of the products' manufacturers and asking about my desired customization. Most of them are pretty amenable to doing custom work. I didn't even provide them with CAD or schematics; I simply explained in words what I wanted them to do, and they drew up a CAD model and produced a single looks-like, works-like prototype for approval before making as many as I wanted. For a small kitchen timer-type device with a custom enclosure, I was charged a flat fee of about $1,000 for the work and the first single prototype.

If the look of your device is as important as the function, but you don't have the time, money, or skill to build something that both looks *and* works like you want, you can split the prototype into two devices. These are commonly referred to as the *looks-like* and the *works-like* prototypes. A looks-like prototype isn't functional but simply

approximates the intended appearance of your product. It can be made with materials like foam-core, 3D printing, or urethane casting. Foam-core and 3D printing are usually the cheapest and fastest ways to get close to the look you want. You may not be able to physically fit your electronics inside this enclosure, but that's where the works-like prototype comes in. It has the functionality of your product, without the polished appearance. The idea is that you can show these two prototypes side-by-side to investors or early users, to demonstrate how the device works while also letting them see the future vision.

Refining Your Prototype

Continue to iterate on your MVP until your users are happy and think you have a viable product. Don't wait for them to stop making feature suggestions, or for every one of your early users to agree on what to do next or that everything is perfect. Your users will never stop making feature suggestions, and you'll feel yourself continually coming up with new features to add, too. On top of that, your users aren't all going to agree with you or with each other. That's okay. The goal is to get to the point where you can stop making major changes to your product and start honing the design in an effort to get an imperfect Version 1 out the door and into more users' hands.

It's impractical to get a design perfect before you send it for fabrication, but how long do you spend trying to find mistakes before you release it? A good rule of thumb is to get to the point where you're about 90 percent confident that the design will work and then release it. Past this point (for most designs), the extra time you'll spend ensuring that everything is 100 percent perfect will take longer than it would to just fix any small mistakes you missed. This won't work if your design review missed something major, so make sure your design reviews contain diverse engineers from multiple groups.

Once you're more sure of what you need to build, you can begin cost engineering. When designers start cost engineering before this point, they raise the likelihood of producing a product that people will feel is cheap. You also risk not meeting all of your product requirements and specifications. We looked at cost engineering in detail in Chapter 9.

Next, you can finally put more effort into industrial design. I've seen entire companies fail because they prioritized the industrial design over meeting the product requirements and building something that works. It sounds crazy, and it doesn't happen in an instant. It's a slippery slope, a boiling-the-frog scenario. You begin by getting an industrial designer to give you several options, and you eventually whittle it down to the one you want. The industrial designer assures you that what they've come up with is definitely manufacturable and should be no problem. You have trouble initially with some manufacturing problems, but they seem solvable. Then, before you know it, you're a couple months behind schedule, but you've already put so much time and money into this design (you went with the really prestigious industrial designer who wasn't cheap, didn't you?), and if you can only figure out a couple more issues, it'll all be fine. Meanwhile, you're burning through money and you miss the schedule to ship by Christmas, and before you know it, you're dead.

You can do another round of cost engineering after industrial design. A good designer will know how to cost-engineer their design. If they don't have a *lot* of experience doing manufacturing, do a round of design for manufacturing (DFM) after industrial design, too. Your factory can help you with this, and they'll be better at it than your industrial designers simply because they know their machines and use them all day, every day.

Intellectual Property Protection

Depending on the kind of person you are, you may feel tempted to take great lengths to protect your intellectual property (IP) and keep others from stealing your idea as you iterate on your MVP. In the past, startups concerned about this have had early users sign NDAs or even refrained from letting anyone interact with the product outside of their company lab. It's true that it's easier to steal IP from a cobbled-together prototype than from a security-hardened final product. In reality, however, this basically never happens. First of all, it's highly unlikely that any of your first users will be motivated to try to rip you off. Second of all, at an early stage, not only will your product be underdeveloped, but the idea

won't have even been fully proven yet. That's the whole reason you're building prototypes!

I've known many early-stage startups that built hacky prototypes and thrust them into the world with zero regard for a competitor coming along and stealing the idea or technology, and none of them had any problems. The truth is, it's cheaper for a big company to wait for you to come up with a good solution and then just try to acquire you early, instead of stealing and replicating your project. This isn't to say that you should never consider security and IP protection; it's just that you shouldn't spend too much time on it when you're in the prototyping stage.

How to Speed Up Hardware Development

Speeding up hardware deployment is really a question of logistics (and possibly manufacturing), and it is outside the scope of this book. Appendix B has a list of logistics and shipping companies that can help you get your device into the hands of your customers as quickly as possible. Some of them can also handle the things inexperienced hardware startups might forget about, like returns, customs, packaging, and warehousing. If you're designing an electronic device, your core business is probably not shipping and logistics, so find a partner company that has a lot of experience and can do it for you!

What *is* within the scope of this book is speeding up "compilation" time, which you can also think of as reducing iteration time. To do this, you need to make a trade-off during prototyping between the speed of development and the reliability of your system. Software developers do this, too, but they can trade reliability for speed more safely and easily because they can deploy almost instantly, meaning that fixing mistakes and improving reliability is cheap. If you want to speed up the iteration time of your hardware development, you'll likely have to sacrifice some reliability. This is okay. Rather than deploy each iteration (like in software), you iterate on less reliable prototypes in your lab before deploying a single reliable design that incorporates all of your prototype

iterations at once. The important thing is not to try to optimize for both reliability and speed at the same time.

What follows are some practical tips for speeding up hardware development. Note that most of these tips are aimed at early development and proving out ideas and basic functionality.

Overnight everything. The cost of waiting an extra day or two is way more than the cost of overnight shipping. Never be waiting on any parts.

Use the fastest turn time possible. The cost of waiting an extra day or two for fabrication or assembly is way more than the cost of expediting turn time.

Cheat. Mock things up using "wrong" hardware (for example, a smart-phone stuffed in a 3D-printed enclosure running an app instead of a custom PCB with a microcontroller and screen). Do things that don't scale. During prototyping, implementation doesn't matter as long as product requirements are met. This isn't to say you should go into production with an expensive, hacky design, but for development and alpha testing with users, cheat as much as you can.

Don't reinvent the wheel. Many engineers feel that the more difficult their work is, the more original it is, and the more clever it is, the better. They feel justified and proud of solving a hard problem. In the real world, however, people don't care about how hard you worked to build something; they only care about the result. Take shortcuts to speed up early-stage development and get help where you can. Use application notes. If you can find a chip that does or almost does what you want, buy it, don't build it.

Get a circuit mill. These can be expensive, with prices ranging from the $4,000 Bantam Tools desktop PCB milling machine to the $25,000 LPKF mill or the LPKF laser machines that are upward of $100,000. The machines can also be expensive to run, since you need to buy lots of consumable end mills that wear out quickly because they're so small, and they come with other constraints, too, like lack of solder mask and the ability to only mill two-layer boards.

Still, getting a circuit mill can dramatically speed up hardware iteration time and end up paying for itself faster than you might think. They're especially valuable for RF development, where PCB manufacturers charge a lot to use exotic substrates and have long lead times for ordering the material they don't have in stock. My company designs a lot of RF equipment like antennas and amplifiers. We can order samples of substrates from Rogers for free and mill out and test designs ourselves within hours. We often mill multiple boards per day. We use the LPKF ProConduct system to make vias, which consists of a silver epoxy that gets sucked through the via holes and baked out to make them conductive. Most of the designs we mill are exclusively surface mount, and our smallest trace/space is 10 mils.

Hire a technician. An experienced technician will let you parallelize work. They're probably faster and more skilled at assembly and rework than you. A great technician is really valuable, and they've gotten me out of jams on multiple occasions.

Hire a PCB design engineer. You may have heard this role referred to as a *layout designer*, but given the amount of actual engineering that goes into modern layout design, it's more accurate to say *PCB design engineer*. The reasoning here is similar to why you should hire a technician: A good, experienced PCB designer is probably going to be better and faster at layout than you are, and they'll let you parallelize work. You should expect to work closely with this person on a daily basis to guide and direct their layout. You could also contract out the layout design work, but this can actually cause delays in some cases. It's hard to closely collaborate with a remote contractor, especially if they're in another time zone. The other issue with a contractor is that they may have more than one job going at a time, which can slow them down. Many people don't realize the effect that a good or bad PCB designer can have on a product and its schedule. Good PCB designers very often have engineering knowledge that many electrical engineers don't, and if hiring the right PCB designer is going to be the difference between

the product passing FCC testing or not, you should really invest the time and effort in finding the right person.

Use the right tools. High-quality, purpose-built tools do a high-quality job and save you time and money in the long run. That includes good CAD tools, good lab equipment, and good components and materials. Accuracy and precision matter, so invest in good tools, even if they're more expensive. Good doesn't necessarily mean expensive, though. Good but cheap test equipment can be had on eBay for an order of magnitude less than brand-new equipment.

Use the simplest architecture possible. Do this even if it means an expensive BOM. There will be fewer things that can go wrong. People don't doubt you can make electronics cheaper; they doubt that the product will work or that people will want it.

Use evaluation and development boards liberally. Hook a bunch together to make prototypes. Design as few custom boards as possible. It will feel like cheating, but that's okay.

Build multiple, separate modules. Use multiple PCBs at first. Make products in chunks instead of as a single huge, integrated design. Have alternates for each module. They can be brought up in parallel and replaced if they break. Redesigns of subsystems are cheaper because you don't need to fab a huge board.

Ignore form until you can't. Make the prototype as ugly as necessary to work. People who really have the problem you're trying to solve won't care about what the solution looks like. At the beginning, having a looks-like model only matters for investors. A works-like model is what you need to figure out product market fit.

Know when to ditch a design. Don't fall in love with any particular implementation. If you start to think of your design as your "child," you risk pursuing it for your ego rather than its quality as a solution.

Pay for an assembly house to build your boards. The only time you shouldn't do this is if assembling the boards yourself would take

less than a day. Usually, it's cheaper to pay for assembly than for you to do it. You can then bring up other boards.

Implement as few features as possible. What's the minimum viable design? Ten different designs with one feature each is better than one design with ten features. This isn't to say that you shouldn't design your hardware to be adaptable, but don't get distracted designing and debugging features that are secondary to the primary problem your device is trying to solve. Don't build a Swiss army knife, build a steak knife.

Divide and conquer. Break up hard problems into multiple smaller problems. Break up systems into subsystems. Take it one part at a time.

Keep a list of chips you like. You should know how well they actually work versus what the datasheet says. This is especially useful for chips that interact with firmware, since you can reuse the hardware abstraction layer and know that it works. You can also reuse your "application circuit," which includes all of the other supporting components needed to make it work.

Use a highly integrated chip for a complex but common task. For example, an Ethernet controller can be implemented using an FPGA and an external PHY, but using a dedicated IC that does everything for you and can just be dropped in is better.

Don't use a highly integrated chip for a simple task. For example, if you need to measure current, don't use a gas gauge chip that talks over SPI and I2C and measures 10 other things. It will take longer to integrate, can add unnecessary dependencies, and usually costs more on your BOM.

Document. Write well. Don't use more words than you need.

Question assumptions. Don't do something just because that's how it's done. On the other hand, don't discount experience or proven components and methods just to be contrarian. If something is always done a certain way, take a step back and ask yourself why.

Simple is better. Ask "do we need that?" for every feature. This doesn't necessarily apply to components on the PCB itself. Getting something working is more important than minimizing component count. You may not need that TVS diode, but you might as well put it in for the first prototype. You can always mark it as do not populate (DNP) later for free.

Don't wait until you have all of the information you need. You'll never have perfect information, and it will always be a little bit incomplete. You'll have to start analyzing, designing, or prototyping anyway. Guess or approximate.

Remember, these tips are only for the prototyping stage. They often sacrifice money and reliability for speed. These sacrifices are okay to make when you're trying to figure out what to build, but they're not okay when you're going to production.

Keep in mind also that while working quickly has its benefits, your pace of development still needs to be sustainable. People won't be able to work 16 hours a day for very long. You'll get high turnover and no one will have a healthy work-life balance. You also need to make sure you don't start to get sloppy. The pressure to move fast makes it tempting to accumulate unsustainable amounts of technical debt, but it's only going to slow down your progress later.

You can decide how much of your prototype you use for your production design. You can use almost all of your prototype code and hardware for production, or you can throw it all away and re-implement everything from scratch. The best option may be somewhere in the middle. Use your judgment and your product sense to determine where it's okay to keep some technical debt and where you need to spend the time up front to redesign.

Helpful Tools

Hardware prototyping has gotten a lot easier in the last 20 years thanks to the growth of the hardware hacker community. There are now myriad platforms, development kits, and communities that can save you

time when prototyping a new device. Let's discuss some ways you can use them successfully.

Modules, Evaluation Boards, and Breakout Boards

Modules, evaluation boards, and breakout boards are a great way to get familiar with a chip or subsystem. Modules and evaluation boards usually include a chip and all of the required circuitry to drive it, while a breakout board usually refers to just a chip mounted on a PCB that makes it easier to solder to. Companies like SparkFun and Adafruit sell dozens of modules and breakout boards for sensors, switching regulators, displays, and more. Having a box full of these modules can allow you to try new ideas or make last-minute modifications very quickly.

Modules and evaluation boards usually have schematics and layouts available and are a great resource for implementing the device on your custom PCB, since you have a known working design to start with. This eliminates the question of whether you soldered something correctly or got the pinout right, or made any other implementation mistake. A prototype made with these devices won't be pretty, and it'll take up much more space than the final version of your product, but you can at least be relatively sure that each module will work on its own.

The main thing to be careful with when using modules is to make sure you have them powered correctly and that all the logic levels between different modules are correct. It's very common for a design to use both 3.3 V and 5 V parts. There are modules available that do bidirectional level conversion between 3.3 V and 5 V. It's good to have a handful of those laying around to use between other modules that need level conversion.

All of these different functional blocks are usually connected with a nest of wire. One way to reduce the amount of soldering necessary while still allowing you to quickly change the connections between different boards is to use Wago lever nuts. Wago is the name of the company that makes them, and they're small terminal blocks that allow you to quickly connect and disconnect wire using a small lever actuator.

A handful of these can be really useful in your toolbox. You can find a link to buy them on this book's website.

Breadboards

Breadboards can be useful for some types of electronics prototyping but not all. They have high stray capacitance and inductance, making them useless for any high-frequency work. As Philip C.D. Hobbs puts it in his book *Building Electro-Optical Systems* (Wiley, 2000), "At least resist using white protoboards above 50 kHz, 100 mA, 50 V, or when a randomly sprinkled 0.1 ohms or 100 nH in series or 10 pF and 10 megaohm in shunt will screw you up."

One simple way breadboards defeat electronics newbies is by putting a break in the power rails on the edges of the board. To make it easier to use multiple voltages on a single breadboard, some breadboards divide the power rails into four quadrants by disconnecting the rails at the middle of the board. Use a multimeter to check continuity between the very top power pins and the very bottom ones. If there's no connection, you'll need to add small jumpers to complete the connections, as shown in Figure 10-1.

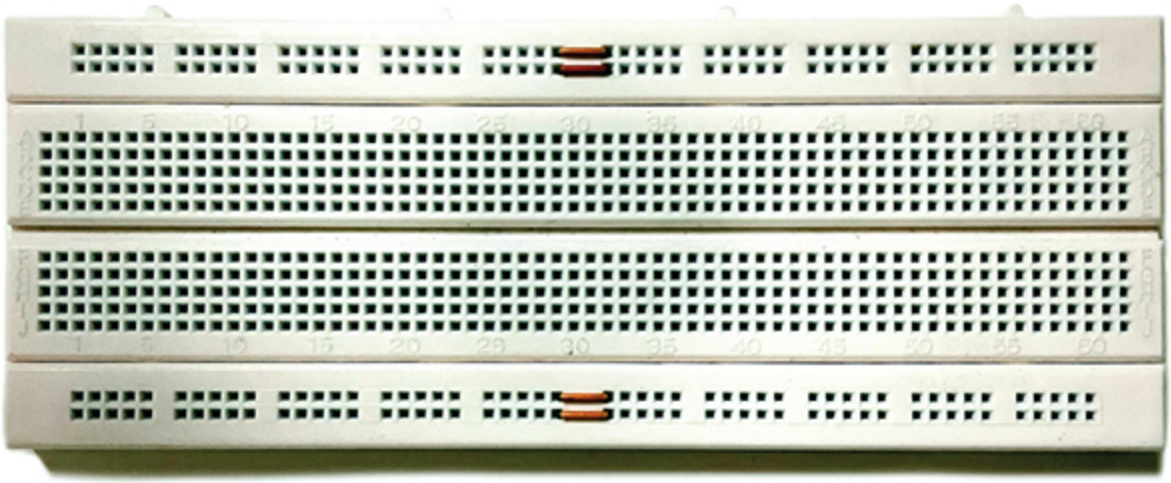


Figure 10-1: Notice the jumper wires to extend the power rails along the entire length of the breadboard.

I usually leave these jumpers in if they're necessary so I don't forget to put them in each time I build a circuit. I've seen several instances

where someone has spent a long time trying to debug a breadboarded circuit, only to discover that the breadboard they were using needed jumpers on the power rails.

Breadboards can also sometimes have problems making a good connection to the wires you stick in the holes. "You had one job!" you'll scream at the breadboard after discovering that this has happened to you. If I have any problems getting a breadboarded circuit to work, I always start by getting out my multimeter and doing continuity checks to make sure everything is DC connected as it should be. The other common mistake this will catch is being off by one hole when you plugged in a jumper wire. Breadboards are quite cheap, so if you're in doubt about whether the contacts are worn out or if one row has a spotty connection, just throw the breadboard away and get a new one. It's not worth wasting your time. I have the same policy with cables and connectors.

Prototyping Platforms

The next step up from prototyping with modules is using purpose-built prototyping platforms. There are a many such platforms available, with more being released all the time. We'll only cover the most popular ones here, like Arduino and Raspberry Pi.

It's important to note that while these platforms can be great for proving out an idea, it's not a good idea to try to take a product to production using them. Prototyping platform boards are designed to be cheap, and they will usually fail EMI and EMC testing. If you do decide to use a version or subsection of a prototyping platform in your product, you'll need to redesign it to ensure that it passes.

Arduino

A lot of more senior engineers may scoff at the idea of using Arduino for any serious prototyping, but don't let that stop you. Arduino has done more for rapid electronics prototyping and electronics education than perhaps any other platform ever. Arduino is an excellent platform for trying things out quickly, and they even offer a line of modules

intended for use in volume production. There are dozens of different sizes and shapes of Arduinos and Arduino-compatible boards that you can choose from. Some contain only the bare minimum required to run code, and some are specifically designed for certain applications. For example, you can find boards tailored to MIDI, automotive applications, wireless sensors, wearables, and many more uses.

One advantage of the Arduino platform is that there's already a wealth of libraries and software written for it, which you can use to jump-start your development. Also, because Arduino is really just a bootloader, you can use the Arduino development environment on lots of different chips with different architectures, like the STM32 family or the more power-efficient MSP430 family. Ultimately, it's possible to deploy a product that runs Arduino code, but for a more reliable and feature-complete codebase, you'll need to develop in an embedded C environment instead. You can often reuse parts of the Arduino libraries, though, since they're written in C.

That said, be careful of software licensing issues. Some Arduino code is distributed under the GPL license, some under the LGPL license, and libraries written by individuals can be under any number of other licenses. Each of these has different requirements, but mostly they specify whether a product that uses parts of that code must be open source. For example, you may use GPL code only if your entire project is *also* GPL licensed (that is, open source). There's a good chance you don't want your product to be open source. Not all software licenses require this, though, and it can be difficult to understand the complex entanglement of different licenses from different libraries. Usually, these rules kick in only when code is *distributed*, so it's still okay to hack together prototypes with Arduino code. The best thing to do is to avoid using open source code in your product without consulting with a lawyer first.

Extra modules designed to work with Arduinos are called *shields*. They're easy to use because they plug straight into the Arduino, meaning they don't require soldering, and they make it impossible to plug in incorrectly. Arduino shields don't have to be used with only Arduinos, so if you're looking for a prototyping module and find one

you like that's actually an Arduino shield, you can easily use it with other prototyping platforms or development boards. You'll just need to wire it together yourself instead of simply plugging it in.

Raspberry Pi

The Raspberry Pi was one of the first extremely low-cost single-board computers on the market. Since its release, it's come quite a long way. The latest version of the Raspberry Pi as of this writing is the Raspberry Pi 5, which has a quad-core 64-bit ARM processor running at 2.4 GHz, with dual-band Wi-Fi, up to 16 GB of RAM, Bluetooth, gigabit Ethernet, dual 4K video outputs, and expanded IO capability for a starting price of $50. There are several different variants of the Raspberry Pi, and new models come out roughly every couple of years. The availability of the different models can vary, so make sure you can actually buy the model you want. Older models are discontinued, and even the latest model can be difficult to find since demand is so high.

Raspberry Pis can run several different flavors of Linux. They also have 27 GPIO pins available on a 40-pin header, which includes SPI, I2C, and UART buses. Like Arduino, the Raspberry Pi has a massive community of developers, with hundreds of libraries available for different applications and peripherals. There are also third-party hardware add-ons like Arduino shields, but they're called HATs (Hardware Attached on Top) in Raspberry Pi parlance.

Raspberry Pi also makes the Raspberry Pi Compute module, which is designed specifically for production use. It's a small PCB with connectors that you can socket into your designs without requiring your board to have a complex HDI stackup or difficult routing. The compute module contains everything you need: the processor, RAM, flash memory, even Wi-Fi and Bluetooth. The disadvantage of using the compute module is that you're at the mercy of the Raspberry Pi supply chain.

While the Raspberry Pi was the first low-cost single-board computer, there are now dozens of competitors at a similar price point. It's hard to enumerate them all, since new boards are released constantly and old boards are commonly discontinued, often with little warning.

Raspberry Pi remains the most popular by far, and competitors often find it difficult to match its price. There are also similar boards to the Raspberry Pi Compute module. They're usually called *system-on-modules*. The Nvidia Jetson is one example, and several processors by NXP are also available in this form.

It's easy to accidentally damage or destroy development boards or single-board computers. Many of them lack protection circuitry, since it adds to the BOM cost. Make sure you don't ever set a GPIO pin as a high output and then short it to ground. You should also never short a GPIO pin set high to another GPIO pin set low. In both of these cases, you can cause a GPIO overcurrent condition. GPIO pins can only source a certain amount of current before they're damaged. This maximum current can be found in the datasheet, and you need to make sure you don't exceed it. Another one of the most common ways to damage one of these boards is to use the wrong logic level on the GPIO pins. If they use 3.3 V logic and you apply a 5 V signal, you'll destroy them.

RF Prototyping Tools

RF prototyping is significantly more difficult than prototyping circuits at lower frequencies. You can't use breadboards because they have too much stray capacitance and inductance. The simplest way to prototype is to use screw-together parts from a company like Mini-Circuits. You can buy amplifiers, attenuators, mixers, splitters, and many other components with SMA connectors. Then it's just a matter of screwing them together. While Mini-Circuits probably has the largest selection of screw-together RF parts, other companies have higher quality or more specialized parts available, too.

X-Microwave offers something closer to an RF-rated breadboard. They produce dozens of small blocks with parts from different manufacturers and have devised a way to fit them all together without any soldering. The fact that they offer so many different parts from different manufacturers sets them apart from other companies that only offer screw-together modules for their own parts. Additionally, they have a free online simulator that you can use to prototype designs in

software first and measure basic parameters before buying all the parts to build it.

Another useful tool for RF prototyping is a software-defined radio (SDR). SDRs allow you to digitally control a radio to transmit or receive over a large frequency range with a fairly large bandwidth. There are many SDRs on the market, but some of the most popular consumer-grade SDRs include the USRP (Universal Serial Radio Peripheral), HackRF, LimeSDR, and bladeRF. These can cost anywhere from several hundred to several thousand dollars, depending on the frequency range and bandwidth you need. A cheaper option for low-bandwidth, low-frequency applications is the RTL-SDR, which can be picked up for about $30. The RTL-SDR is great for simple receive tests or just experimenting, but be aware that this device is receive only, so it can't be used as a transmitter. The RTL-SDR also has an upper frequency range of usually about 1.75 GHz (depending on the exact model you get), and a bandwidth of about 2.5 MHz.

Almost all SDRs can be controlled using a piece of open source software called GNU Radio. There are dozens of packages and plug-ins for GNU Radio that allow you to do very complex things. There's also quite an active development community with lots of good documentation and tutorials online.

NOTE

Prototyping with RF also requires some specialized test equipment to take measurements. See Chapter 11 for more information on what equipment you'll need and how to get it.

Maintaining a Lab Notebook

A lab notebook is the cheapest but most important tool in your lab. Robert Pirsig sums up the value of a lab notebook in his classic *Zen and the Art of Motorcycle Maintenance* (1974):

> When you've hit a really tough one, tried everything, racked your brain and nothing works, and you know that this time Nature has really decided to be difficult, you say,

> "Okay, Nature, that's the end of the nice guy," and you crank up the formal scientific method.
>
> For this you keep a lab notebook. Everything gets written down, formally, so that you know at all times where you are, where you've been, where you're going and where you want to get. In scientific work and electronics technology this is necessary because otherwise the problems get so complex you get lost in them and confused and forget what you know and what you don't know and have to give up.

A good lab notebook serves several functions. First, as Pirsig articulates, it's a valuable resource when troubleshooting problems. There's more information about this in Chapter 14, but a lab notebook can keep your ideas and hypotheses organized as you work through the science of troubleshooting. It can also help keep you from accidentally doing the same work twice.

A lab notebook should also be a good enough record of what you tried and designed so that someone else with no prior knowledge of the project can understand, replicate, and contribute to your work using only that notebook. It's not uncommon for a team to gain and lose people as time goes on, so it's very likely that your notebook will be used in this way at some point.

Similarly, a good lab notebook helps prevent knowledge from being siloed. A term sometimes morbidly used to describe this is the *bus factor*, which is the number of people on the team that would need to be hit by a bus to halt development progress due to lost knowledge. If a team has one person, the bus factor is one. If a team has five people, ideally the bus factor would be five because knowledge is spread out widely enough among everyone that there isn't a single point of intellectual failure. Plus, everyone is ideally keeping a detailed record of what they're doing in their lab notebook.

NOTE

If the idea of large numbers of your team being hit by a bus seems unlikely to you, remember that the much less dramatic but all-too-common occurrence of engineers simply leaving the company will have the same effect—at least for you.

The last thing that a notebook is valuable for is protecting you against claims of fraud or other engineering malpractice. If another company sues you, or if there's an investigation about a product's safety, having good documentation about what you did and why during the development process can be valuable evidence in a courtroom, if it ever comes to that. Even in cases where the law isn't involved, such as if upper management is looking for a scapegoat or trying to place blame, you can use your notebook to prove that you acted ethically and in good faith.

Before the America Invents Act was passed, lab notebooks were used as legal documents in the US to prove who came up with a particular idea and when exactly they invented it. Since the US patent system is now first to file instead of first to invent, lab notebooks aren't really used this way in court anymore. While it's probably unnecessary to continue practices like having both the author and a witness sign every page in the notebook, it's still useful to number and date every page.

Notebook Options

Some people prefer a digital notebook because it's easier to search for text and integrate images or other media. Personally, I prefer a physical notebook. It doesn't need to be charged; it won't get accidentally deleted, corrupted, or hacked; and it's less distracting (especially during meetings). You can still include images (and you should) by printing them out and taping or stapling them in the notebook.

The Scientific Notebook Company makes great physical notebooks that I highly recommend. They also sell archival ink pens, which are great for writing in a lab notebook. Archival ink won't fade or bleed, is waterproof, and won't smear or feather. Other good notebook manufacturers are Moleskine, Eureka, National, and Leuchtturm1917. Hybrid digital-physical solutions exist as well, like Rocketbook and smart pens. Popular purely digital solutions include the e-ink notebook called reMarkable, an iPad Pro with a Pencil, and Notion or Google Docs. These tools are nice because you can drop in links to spreadsheets or code, and paste in screenshots, diagrams, or anything else. You can find links to all of these options and more on this book's website.

What a Notebook Should Include

The inside cover of your notebook should have your name, phone number, and address, as well as the notebook's volume number and the dates covered. Some notebooks give you space at the beginning for a table of contents. If your notebook doesn't have that, leave enough space to make one yourself. As you add entries, add them to the table of contents. Write down the name of the experiment or investigation, and the page number where it starts. Don't try to break up your notebooks by topic, just keep everything in chronological order. It'll be easier to find things later.

Each entry should have a title, date, the hypothesis that you're testing or motivation for what you're doing, and a brief summary of any relevant background. The meat of the entry should have the materials you used, the conditions under which the test was performed, the equipment used, the setup of the experiment, the results, an analysis of the results, and a conclusion. Include raw data, screenshots, plots, and any other relevant information. If your data is too big to fit, explain where to find it. If you really want to go the extra mile, you could even tape in a microSD card with the data on it.

Use your judgment to decide how much detail to include in your notebook. There's a trade-off between keeping a good notebook and spending so much time on it that your progress on the actual product is very slow. It's probably a good idea to record the fact that you used high-Q capacitors when tuning an antenna, for example, but it's probably not necessary to specify the mix of solder that you used. You do want to include tests that failed or didn't work as you expect. Negative results are often just as important as positive results. If data doesn't come out looking the way you want, or a plot turns out ugly, keep it in there anyway. It's the honest thing to do, and it will help you figure out whether you need to repeat experiments or conduct them in a different way to improve your results.

For the same reasons, don't remove any pages from your notebook. If you need to cross something out, strike through it with a single line rather than scribbling over it. You might want to read about the mistake you made later. Don't use scratch paper, use your notebook. If you're a

perfectionist, you may be tempted to wait until you've organized your thoughts somewhere else before you make an entry, or to otherwise make something "good enough" to go into the notebook. Don't do this. The lab notebook isn't sacred. It's a place for recording wrong procedures, tests that didn't turn out right, and stupid mistakes, just as it's a place for elegant and beautiful "eureka moment" breakthroughs. What matters is that everything is captured, not that it's a perfect picture of a genius at work. In the words of Ms. Frizzle, "Take chances, make mistakes, get messy."

Some people say you shouldn't treat a lab notebook like a journal, but I disagree. A good notebook can almost act like a mind map, which will help you and others who read your notebook understand your train of thought and your motivations for your designs and experiments. It will help you stay focused, keep things organized, and see the bigger picture. Writing more like a journal will also make it much easier to get back into the same frame of mind after a long break. After not thinking about a particular problem for a few weeks, it's immensely helpful for me to read the "story" of where I was and what I was doing. Again, see Chapter 14 for more information and tips about using this technique for troubleshooting.

It's a good idea to back up your lab notebook. If it's digital, this is easy. Just keep another copy of the files on a thumb drive, ideally geographically distant from where you work. If there's a fire, you shouldn't lose anything. Backing up a physical notebook is more work: You need to photograph or scan every page of your notebook and keep the pictures backed up in the same way as the digital notebook. It's harder to automate this with a physical notebook than a digital notebook, but you should still take the time to do it every once in a while.

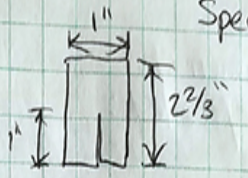
Figures 10-2 and 10-3 show two pages from one of the best lab notebooks I've ever seen. I bought this notebook at a surplus sale, and it appears to be left over from a medical device company that went out of business.

The first page shows all of the best practices I've mentioned: a title, page number, and date; a description of the experiment; a drawing of

the setup; and taped-in data from the experiment. The second image shows an example of taping in pictures of the experimental results.

Tear Test of Saatitech150

GL = 1"

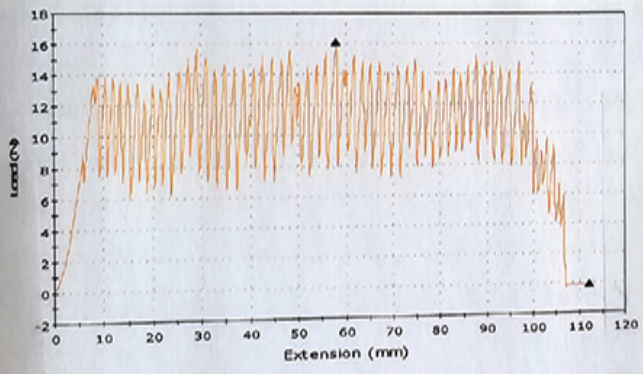


Specimen dimensions for tear test

Instron, Juva 0001 & load cell of 100lb, Juva 0083 were used for this test, see pg 47 for calibration.
Testing was done by EM
Jaws of Instron are in smooth condition for this test. The jaws hold ≈ ½" of fabric of each leg of the test specimen, similar to the specimen seen on page 84.
The maximum load of Saatitech 150 during the tear test was 16.14 N according to Instron output.

Instron Output of Tear Test:

5 Peak Average Method per ASTM D 2261-96:



	Maximum Load (N)	Extension at Break (mm)
Saatitech150	16.14128	112.066

EM 11/20/06

Saatitech 150 Peak a

Peak	Load (N)
1	16.14
2	15.66
3	15.51
4	15.36
5	15.24
Average	15.58
Stdev	0.35
Max	16.14

EM/12/06

From analysis of the tear test data the average of the five maximum loads came to 15.6N for Saatitech150, this value is slightly lower than the instron max load output. Saatitech150 has a comparable tearing strength [illegible] SD60-5 & SD57-5 with 15N and

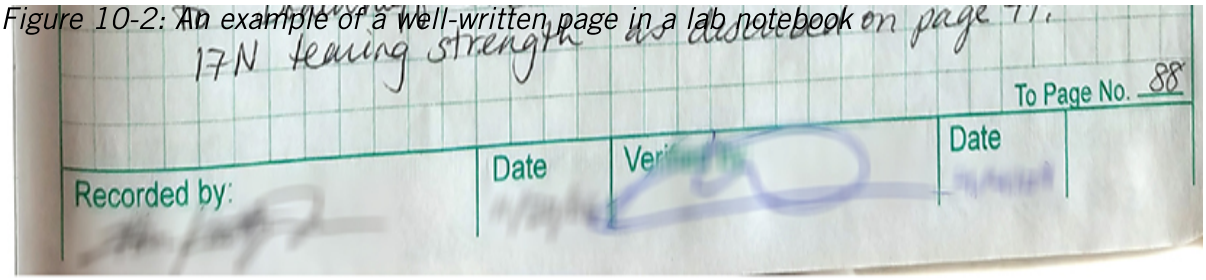


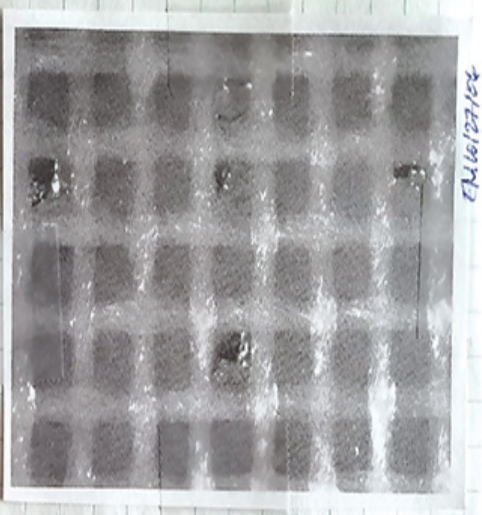
Figure 10-2: An example of a well-written page in a lab notebook

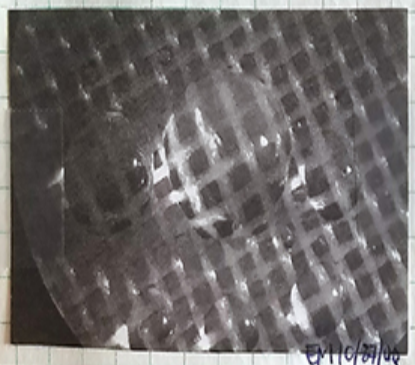
Project No. N/A

Book No. 009 TITLE [illegible]

From Page No. ~~64~~ 65

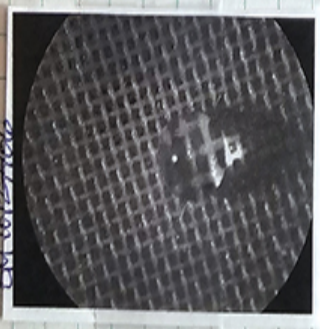
Joanne Fabric with 6640 & 2x4901 that has been stuck 20 times by a 33.5 gage needle. →





← Leak seen immediately after 20 sticks by 33.5 gage needle in Joanne Fabric with 6640 & 2x4901

← One leak seen after cycling Joanne Fabric with 6640 & 2x4901 ~~to~~ from 1 psi to 5 psi.



← One leak seen after cycling Joanne Fabric with 2x4901 from 1 psi to 5 psi

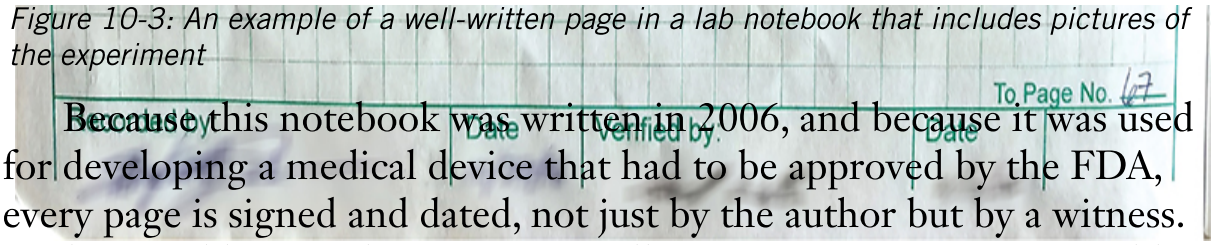


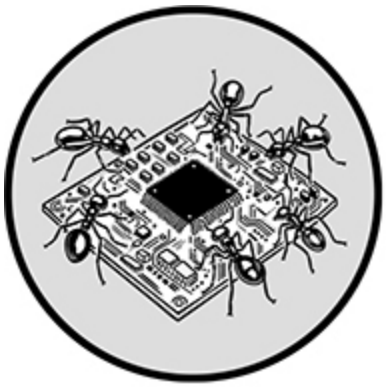
Figure 10-3: An example of a well-written page in a lab notebook that includes pictures of the experiment

Because this notebook was written in 2006, and because it was used for developing a medical device that had to be approved by the FDA, every page is signed and dated, not just by the author but by a witness. As discussed before, this step isn't really necessary anymore. If you like the way this particular notebook is laid out, you can buy a blank one from the Scientific Notebook Company.

Conclusion

A prototype should help you figure out not only *how* to build your product, but more importantly, *what* to build in the first place. Take some time to plan out your prototype development process rather than just blindly soldering things together. Good prototyping can be very fast and teach you a lot very quickly. It will also prevent you from going down the rabbit hole and spending lots of time and money on something that won't work. This is one of the most valuable skills you can develop. I think it's also the most fun part of hardware development, since you get the satisfaction of seeing your idea in real life for the first time, even if it's not in perfect fidelity yet.

11
BUILDING A LAB



Testing and developing electronics requires a lab. This chapter covers the supplies and equipment you'll need to get your work done effectively. You'll find guidance on stocking and maintaining a lab at any budget level.

Before you start buying equipment, you must decide what lab capabilities you need, and that requires making some kind of trade-off between time and money. The fancier the equipment you get, the faster you can prototype and debug (to a point), but this speed can come at a cost of hundreds of thousands of dollars in equipment. Then again, you can build a lab that will get you 80 percent of the performance for much, much less. Even if you're doing RF work that traditionally requires expensive equipment, there are ways you can get by on a fraction of what a well-funded lab would spend.

Lab Safety

It's important to take steps to make your lab safe. The most common ways people get hurt in electronics labs are from cuts and burns. You can buy an off-the-shelf first aid kit and upgrade it with burn gels or hydrogel dressings specifically to treat burns from soldering irons or hot

equipment. You'll also want a good supply of safety glasses for people to grab. If you're using any big power tools, a bottle of earplugs or other hearing protection is important. I also like to keep at least one respirator mask with P100 organic vapor cartridge filters in it for occasions when someone is cutting or heating something that may produce dust or vapors.

An eyewash station is good to have, and if you can't get one plumbed in, you can buy stations that have a built-in saline reservoir and don't need any plumbing. You'll also want a flammables cabinet to store highly flammable chemicals like isopropyl alcohol, acetone, and other solvents. When these chemicals are on the bench in small bottles or dispensers, those containers need to be labeled with the name of the chemical and any hazard symbols. Make sure to keep all material safety datasheets for any chemicals you buy.

There's a higher risk for fire in an electronics lab, particularly if you're working with batteries. Keep a fire extinguisher that's rated for putting out class A, B, and C fires on hand. It's also a good idea to have a metal bucket of sand you can use if a battery catches fire. You can drop the battery in the sand or dump sand on top of the battery. The sand won't put out the fire, but it will help contain the fire. You can also buy lithium battery charging bags that can contain battery fires. Similarly, I like to use charging bags to store batteries that need to be disposed of.

Make sure your flooring is nonslip and that cables are routed or taped down so they're not a tripping hazard. Keeping the lab organized will also help keep people safe. Everything should have a home and make it back there after use. This is so much easier said than done.

Leaded solder is still pretty common, so many people wonder if they should be worried about lead exposure. There's no known safe level of lead for humans, so you should take steps to minimize your exposure to be as low as possible, ideally zero. Contrary to popular belief, leaded solder fumes don't contain significant amounts of lead. The smoke you see is flux vaporizing and breaking down. Lead starts to release significant fumes at about 900°F (482°C). Harmful vapors are created at temperatures above 1800°F (982°C). Some soldering irons can reach a maximum of 932°F (500°C). If you operate one of these soldering irons

as hot as it can possibly go, you may be releasing some lead vapors. If you're worried about it, just keep your soldering iron temperature below 900°F (482°C) when working with leaded solder.

A fume extractor can help remove irritating flux smoke, but they're not very good at removing lead vapor, even if you use a HEPA filter. For that you'll need to use an activated charcoal filter and possibly more specialized equipment. That said, the most likely way you'll be exposed to lead isn't via vapor: It's by ingesting it. Solid lead metal is relatively harmless to touch, but dust or particles can transfer from your fingers into your mouth if you touch your face or eat or drink after soldering without washing your hands. The simplest way to avoid this is to make sure you wash your hands after soldering.

Lab Furniture

The most important furniture in your lab is the lighting. Get ring lights for your microscopes and very bright overhead lights for the lab bench. When you're working with parts that are hundredths of an inch long, you don't want any shadows. You also want the light to be as white as possible to make discolorations of parts easier to see.

Get a lab bench that has shelves above it for test equipment. You can build a lab bench using Ikea furniture, or you can buy a purpose-built one. Most test equipment is very long and deep, so putting it directly on your work surface won't leave you much room for doing anything. A shelf keeps your primary work surface clear and makes it easy to drop down probes from the test equipment to the device you're testing. This can really improve cable management, since instead of cables being sprawled out all over the table, they drop straight down from above and don't get tangled up. You also want to get as deep of a shelf as you can so you can push your test equipment back to a comfortable distance and not have it hanging halfway off the edge.

The work surface itself needs to be ESD-safe and heat-safe. You can buy ESD soldering mats made of a high-temperature-resistant material. ESD mats usually have a connector on them for attaching an ESD wrist band. For this wrist band to be effective, you need to ground the ESD

mat to earth ground. Other ways to manage static include antistatic floor mats and even antistatic carpet.

Simple ergonomics are important. Get a good chair and make sure it's at the right height. Especially if you're working under a microscope a lot, you need to make sure you're sitting up straight and not slouching. You may need to put some books under your microscope if it doesn't adjust high enough.

Electronics assembly often involves the use of solvents like acetone, isopropyl alcohol, and other flammable substances. Get a flammables cabinet to store these materials. Such cabinets aren't very expensive, and they go a long way in keeping you safe. Of course, for the cabinet to be effective, you actually have to use it. A small amount of properly labeled isopropyl in a dispenser on your lab bench is fine, but keep the gallon refill jug in the flammables cabinet. If you're working in a commercial space, it may be against the law or a code violation if you don't.

Soldering Equipment

Many people think soldering is difficult because they've only used low-quality tools. Once you start using a nice iron in good condition, a microscope, and a good set of tweezers, you'll be able to solder small components after a few hours of practice.

Irons

Buying a soldering iron is like buying a mattress or tires. It's worth it to spend more money because you're going to be using it constantly. If you have the budget, Metcal and JBC are the two best manufacturers. Their irons can cost between $500 and $1,000 new. If you can't afford that, you can usually pick up used stations on eBay for much less.

Metcal and JBC products work a little differently than other soldering irons in that the tip itself has the heating element in it. This means the temperature regulation at the tip is much better than a normal iron, but it also means that new tips are more expensive. Both Metcal and JBC use induction heating, so you don't need to wait for the

tips to get hot; they reach their set temperatures within seconds of turning on. Metcal irons don't have a temperature control on the soldering station but use the tip itself to control temperature. You order particular tips that reach particular temperatures. Both Metcal and JBC offer tweezer soldering irons as well. These are great for quickly adding or removing SMT capacitors and inductors.

The tier below Metcal and JBC is Weller and Hakko. Many websites and YouTube videos will recommend a Weller or Hakko as a first soldering iron, and indeed these irons are fine and work well. Temperature control isn't quite as good and you have to wait for them to heat up, but they use traditional tips that are very cheap. People have gone their entire careers soldering on only a Weller iron. Hakko has really upped its game in the last several years and offers irons that some say approach Metcal or JBC in quality. Don't let the Fisher-Price aesthetics fool you. Hakko also offers irons that use induction heating.

The lowest tier of soldering iron is the "Radio Shack Special." These are irons that plug directly into the wall, have no temperature control, and really don't work well. You can often find an old, low-end Metcal iron on eBay for only a little more than these irons, and you'll be much happier with it.

Hot-air irons are a different market than regular soldering irons. The high-end soldering iron companies also make hot-air rework stations, but the majority of the hot-air stations you'll find online are Chinese. There are maybe three or four different designs and probably 20 companies that make them. They often look exactly the same but have a different name on them. The good news is that these stations are fairly cheap, but they can also be hit or miss in terms of reliability. I've seen stations last many years while others last only months. I've had most success with the Hakko and Quick brands. Metcal makes an expensive hot-air soldering station (the HCT-1000) that I bought in an attempt to have one that would actually last, but it's very complicated to use and I don't recommend it. They do make some other hotair stations that seem to be less complicated and may work better, like the HCT2-200. See this book's website for a list of hot-air irons and where to buy them.

Like regular soldering irons, hot-air irons have different tips available. I get the most use out of small, circular nozzles. If you're going to be doing a lot of hot-air work, a board preheater can also be really helpful. This is a device that sits under your PCB and heats up the entire board but not enough to melt the solder. If your PCB is large or has a large thermal mass, it can be difficult to get a hot-air iron to heat up a small area enough for the solder to melt. The preheater raises the temperature of the entire board enough to make this easier.

Microscopes

Your lab will need at least one microscope per soldering station. The term to search for when shopping for one is *inspection microscope*. These are designed to have a much longer focal length than a normal microscope and a lower magnification. The objective lenses shouldn't be more than 10x, and a zoom level beyond about 5x isn't very useful. You also need your microscope to be binocular—that is, to have two eyepieces. Binocular vision gives you a sense of depth, which is immensely helpful when trying to solder anything under magnification.

The cheapest option for an electronics lab microscope is about $200 from Amazon. It has a built-in light, which works well, and the magnification range is good. A step up from that is an articulated-arm microscope. Amscope is a popular brand. These have better optics, larger zoom ranges, and give you more room to work. They also have really handy mounts that let you easily swing or slide the microscope out of the way when you're not using it. This sounds like a minor convenience, but it's actually extremely useful and lets you get more use out of your bench space.

The next step up from a binocular articulated-arm microscope is something like a Mantis microscope. These use special optics that let you look into something more like a screen than two eyepieces. If you're going to spend all day every day over a microscope, this feature is easier on your eyes. These microscopes are much more expensive than an Amscope; a Mantis goes for about $2,000, whereas an Amscope is about $600.

Whatever microscope you use, make sure you adjust the focus on each objective independently before you adjust the height of the microscope, so each eye is in focus. You may not notice at first if one eye is slightly out of focus, but it can give you a headache after a day of working. You should also parfocalize your microscope before you start working. This will allow your view to remain in focus even if you adjust the zoom level. To do this, first put your workpiece under the microscope and zoom all the way in. Then adjust the focus of the microscope until the piece is in focus. Next, zoom all the way back out and use the diopter rings to adjust each eye until they're in focus again. The diopter rings are located on the tube that the objective lenses slide into. Once they're adjusted, you should be able to zoom all the way in and out with the subject remaining in focus the whole time.

Other Equipment

As mentioned earlier, a fume extractor or fan is good for keeping solder smoke away from your face. Again, most soldering irons can't reach high enough temperatures to vaporize lead, so the smoke is almost all flux. It's not poisonous, but like any smoke, it's not great to breathe it in all the time. It's also just annoying to have smoke in your eyes and face. A simple fan works well, but high-end systems use a vacuum with a flexible tube that you can position next to your workpiece to suck the fumes away and avoid blowing fumes at your neighbor.

Every lab should have a vise to hold PCBs and other small workpieces. Most people go with Panavise products since they are specifically designed to hold a PCB. You can also use a "third hand." The most commonly available type has two alligator clips attached to flexible arms and a magnifying glass attached to a third flexible arm. You can use the alligator clips to hold and position wires and PCBs while you work on them. This is really useful for positioning objects in weird ways that a normal vise won't allow. Unfortunately, most third hands, like the one shown in Figure 11-1, are really terrible and cheap. The alligator clips break off the arms easily, the magnifying glass is basically useless, and the base is too small, so the whole thing readily tips over.

Figure 11-1: A terrible third hand

Luckily, you have a few options for a better third hand. You can find 3D-printable designs on Thingiverse that have as many as eight arms, including some with built-in minigrabbers for easy probing. One design on Thingi-verse that I particularly like is called PCB Workstation Kit with Articulated Arms by giufini. You can print his models for free or pay him for already printed parts.

Another option for a third hand is to build one using plastic flexible segmented coolant hose. There are several tutorials available online on how to do this, but once you get the gist of how the construction works, it's easy to customize it for your own purposes.

All that said, my new favorite third hand is a product called Omnifixo (Figure 11-2). It has magnetic ball joints that allow you to put the spring-loaded clamps in almost any position. I really like how much freedom of movement it allows, and the style of clamp it uses to hold your workpiece is much better than alligator clips.

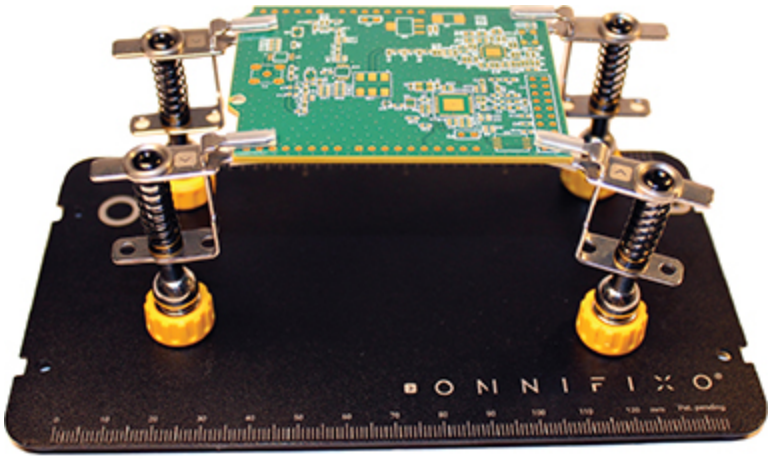


Figure 11-2: An Omnifixo third hand makes it easy to hold PCBs or wires in almost any position.

If you're applying solder paste with a syringe, I highly recommend using a pneumatic dispenser (see Chapter 12 for more). It's much easier on your hand and thumb and allows you to apply paste more precisely. To do this, you'll need a dispenser regulator. These are often marketed as solder paste dispensers, glue dispensers, or liquid dispensers—they're all the same thing. The device has a pressure gauge, a regulator knob, an air hose input on the back, and at least one air hose output on the front. You'll need an air compressor to run it, and possibly some hose fittings. Try to get a quiet air compressor (I like the Ultra Quiet series from California Air Tools), or at least keep the compressor in a soundproof closet somewhere rather than in the lab.

Typically, you'll set the air compressor pressure to 100 PSI or so and run that to the back of the solder paste dispenser. Then remove the plunger from your solder paste syringe and clip on a sealed connection to the output air hose from the dispenser. Make sure you have the right syringe size for the connector to your regulator. Typical sizes are 5 cc and 10 cc. Next, use the regulator knob on the dispenser to adjust the speed at which paste comes out of your syringe. Many dispensers come with a foot pedal to control when paste comes out.

Some solder paste dispensers have DIP switches on the back to change the mode of operation from continuous flow (pressure is applied to the syringe as long as the foot pedal is depressed) to discrete flow (pressure is applied for a fixed time interval once the foot pedal is

depressed). You'll probably want to use continuous flow. You can find a list of recommended dispensers, parts, and fittings on this book's website.

Lab Storage Options

As you accumulate parts, you're going to need a place to store them and index them to help keep your lab organized. The solution most people use is component drawers. These work pretty well, though they can take up a lot of space if you have a wide range of unique parts. If you do use component drawers, get ones that are clear so you can see what's inside them. They should also click shut if possible, so you know they're closed and that they'll stay closed on their own. It's also nice if the back of the drawer is completely enclosed, so if you flip the drawers upside down or lay them flat, parts won't spill between drawers. You'll appreciate this if you ever have to move your component drawers between buildings.

Component drawers aren't the only storage option. For example, another way to organize parts is in reels. This works best for components that you have hundreds or thousands of, but you can also buy cut tape mounted in reels with smaller quantities. Reels can be easier to store, since you can just use a bookshelf or something similar. Alternatively, a company called AideTek makes some really cool folding boxes with tons of little snapping compartments for holding loose SMT parts. Yet another option is to use part notebooks: You can buy three-ring binders with plastic sleeves in them that fit cut tape sizes, as shown in Figure 11-3.

Figure 11-3: A binder for holding cut tape

If you use a binder for holding cut tape, you'll quickly run into the problem of needing to insert a new piece of cut tape into the middle of a page that's already full. To make moving the cut tape easier, don't use stickers to label each entry. In Figure 11-3, each row of components has a label right above it. These labels were made by cutting the DigiKey label that the components came with into a rectangle containing only the DigiKey part number and the part value and description. These two rows of text happen to be right next to one another, so you can fold the cut-out label in half longways (without removing the sticker backing) and slide it in the sleeve. That way, the component values are printed right above the parts, and the back of the sleeve has the exact DigiKey part number in case you want to order more. Making your own binders this way is vastly superior to having a box somewhere filled with bags of parts that you have to dig through when you want to find a particular part. It really doesn't take that long to fill up a couple of binders, and it will save you time, space, and money.

If you don't want to make your own parts books, or if you have too many gaps in your parts collection, you can buy premade books of

resistors, capacitors, and inductors. Sometimes companies will even give you these for free. I like to have books with 0402 and 0603 sizes of each. I also like to keep a book of 0402 high-Q inductors and capacitors for RF work.

Electronic components aren't the only things that come in preorganized sets. It's really nice to have a couple of fastener sets with imperial and metric nuts, bolts, washers, and spacers. Having both metal and nylon fasteners available will give you even more flexibility. Heat-shrink tubing of different diameters is also available in small sets and is equally useful.

Some people insist that everything you use to store a component be antistatic. Static typically isn't a problem because there isn't enough capacitance between the component leads and another object with a significantly lower potential (aka, ground). However, this may not be the case for parts on a PCB, because they will have a significantly higher capacitance to ground. If you're worried about your components, or want to play it safe, just keep components in antistatic bags if your component drawers or other storage system is not antistatic. For example, large, gallon-sized antistatic bags are great for holding old project parts: When I'm done with a project (or if I'm putting it on ice for a while), I put all the parts, PCBs, and notes in one of these bags, put a piece of tape labeled with the project name on the outside, and stick it in a filing cabinet. Smaller antistatic plastic bags are also useful for storing individually populated PCBs. You can buy all sizes of antistatic bags very cheaply, but I can usually keep my small-sized stock full by just saving the extra bags that come when I order parts.

If you want to be *really* organized, there's open source inventory-management software you can use to keep track of where parts are and how many you have. It can be really difficult to keep inventories updated, especially when you're hacking around on something or building a quick prototype. However, for large-scale operations (like a company), inventory management becomes necessary at some point.

One last tip: Small petri dishes are excellent for holding screws and parts that you're taking off or putting onto an assembly. I like them because I can put some parts in a dish, label the dish's lid with a Sharpie,

and then stack several dishes up near my microscope. When I need a part, or if I'm going back to work on something, it's really easy to find. Try to get glass petri dishes if possible; it's easy to accidentally melt plastic dishes. Get half-size dishes instead of the normal full size.

Test Equipment

Having reliable, accurate test equipment is crucial to testing your device. We'll cover the testing process in detail in Chapter 13. For now, this section discusses where and how to get the appropriate test equipment for your lab.

The biggest test equipment companies are Keysight (formerly known as Agilent, formerly known as HP), Rohde & Schwarz, and Tektronix. They've been around a long time and make high-quality equipment, but they're also the most expensive. However, they're far from the *only* test equipment companies: There are plenty of others of varying quality with different specializations. Other names to look up include Keithley (now owned by Tektronix), Fluke, B&K Precision, GW Instek, Siglent, Rigol, and Anritsu. I'll call out certain models that I like in this section, but remember that all of these companies make competing equipment, and it's worth it to shop around between them. If you work for a company or startup, contact the sales engineers at these companies and they'll usually let you try out the equipment for a week or two before you buy it.

Test equipment is notorious for being very expensive. To save money, ask for quotes on used or demo equipment. This is usually equipment that's been loaned to different companies to try out or to borrow while the units they bought were being repaired or calibrated. Stock of demo equipment is often limited, so you may not be able to get the exact models you want. Another way to save money is to buy the previous model. New versions of test equipment don't come out every year, but when the transition happens, you can sometimes get the previous version for less than the new version. Other times, the manufacturer will purposely price the new version slightly lower than the old version to incentivize people to buy the new version.

You can also try negotiating down the price on new equipment. This works best if you have multiple companies fighting for your business. A nice technique to use is to make sure one company visits your lab and sees their competitors' equipment being evaluated on your bench. Sales engineers have some flexibility in the discounts they can offer, and it's always worth asking. But don't negotiate so hard that you burn your relationship with your sales engineers. They're fantastic assets. The representative for my company has gone above and beyond multiple times for us, including letting us borrow very expensive equipment for free and spending time helping us figure out our specific test and measurement problems.

When looking for used equipment online, make sure to research the instrument before you buy it. Different options may be enabled or disabled, and features you were expecting may be missing. The best way to do this research is to search for the instrument's datasheet or manual. These are often old scanned documents that you have to dig through to find the information you want. Pay careful attention to what options and variants of the instrument are available, and check the details of the listing you're looking to buy to verify it has everything you need. Unfortunately, there isn't one central location for documentation of all old test equipment, although the Tek wiki (*http://w140.com/tekwiki/*) is a great resource for old Tektronix gear.

Test equipment is often overbuilt and lasts a very long time. We run equipment in my lab that's older than I am, and it does whatever we ask of it every day. Because of how robust many instruments are, they change hands throughout their lives through sites like eBay and steadily drop in price. Keysight even runs their own official eBay store to sell used units. Besides eBay, you can ask test equipment rental companies about their old units, check out ham radio swap meets, or watch for auctions from old military labs, university surplus, and bankrupt tech companies.

Power Supplies

If you need a really cheap power supply, search for *DPS adjustable power supply modules* on eBay. These are small benchtop power supply "front

ends" that cost about $40. You need to supply a fixed voltage and current, but the DPS turns that into an adjustable power supply with a color LCD screen, current limiting, and power limiting. It's not going to have high accuracy, but it's really not a bad little power supply. There are several models available with different max voltage and current limits. Search for terms like *DP20V2A* or *DPS3003* to find specific ones.

Another nice Chinese gadget to have on your desk is something known only on eBay as a *component tester*. This is a little LCD screen that runs on a 9 V battery and measures various parameters of transistors, MOSFETs, diodes, capacitors, inductors, and resistors. You can find them for about $15. Again, no guarantees on accuracy, but it's really handy to be able to quickly identify all loose components you've got lying around.

Nicer benchtop power supplies come from companies like Keithley, B&K Precision, and Keysight. These start in the mid-hundreds of dollars and end, as usual, in the thousands. They're very accurate and most come with extensive programming and automation abilities. This is great for when you want to set up an automated testing rig and need to control your bench-top supplies in software. To save money, look for supplies with multiple independent channels. Instead of buying two power supplies with one output each, you can buy a single supply with two outputs.

Multimeters

Multimeters are handy for measuring voltage, current, continuity, resistance, and sometimes capacitance and diode forward voltage. They come in either handheld or benchtop versions. A multimeter should be the first piece of test equipment that you buy. There are many cheap multimeters that you can buy online, but they're typically inaccurate and unreliable. Buy a Fluke instead. The Fluke 115 is a great first multimeter. It's about $230, accurate (and remains calibrated for a long time), and very rugged and reliable. If you want something cheaper, check out the Fluke 101, which can be found for about $55. It's a much better value than the plethora of unreliable $20 multimeters you'll find.

Keysight makes great benchtop multimeters. While they're not very portable and must be plugged into the wall, they offer much higher precision than a handheld multimeter. They're also capable of taking measurements programmatically and remotely. Some models have the ability to measure down to picoamp resolution, which is really useful when characterizing low-power systems.

When buying multimeters, you'll often see the display specified in half digits, as in "5 1/2-digit display." This refers to digits that don't have a full range of 0 to 9. For example, a 5 1/2-digit resolution would mean that the display has five digits that can range from 0 to 9, and one digit that ranges from 0 to 1. The "1/2" means the digit has a maximum value of 1 and two possible values (0 and 1). If you had a "4/5" digit, that would mean the digit could display a maximum value of 4, with five possible values (0, 1, 2, 3, and 4). Pretty much all multimeters will only use half digits to express range. Remember, though, that display digits aren't the same thing as resolution. The thing you really want to look for when buying a multimeter is resolution and full-scale voltage range, not necessarily how many display digits there are.

Oscilloscopes

After a multimeter, the first big test equipment purchase a lot of people make is an oscilloscope. There are lots of factors to consider when buying an oscilloscope, but bandwidth is one of the most important. As a rule of thumb, buy a scope that has a bandwidth at least three times higher than the highest frequency signal you want to measure. This may seem confusing, especially if you have a tight budget. Why can't a 1 GHz scope see a 1 GHz signal? Well, it can, but when you're measuring a 1 GHz signal, you actually need to be able to see frequencies above that. If you can see up to the third harmonic of the signal of interest, you'll have a much better view of the true shape of the wave, especially its edges.

You can find oscilloscopes in all price ranges, from under $100 to tens of thousands of dollars. Rigol is a popular entry-level brand, with their DS1000Z series models as low as $280. This will get you two channels and a bandwidth of 100 MHz. Rigol makes higher-end scopes,

too, with much lower prices than comparable models from Keysight or Rohde & Schwarz. I haven't tested these higher-end models, so I can't vouch for them, but it may be worth a look if you have no other option.

Tektronix has some low-end scopes that start at about $600, but the company has received the most acclaim for its MSO series. The MSO22 starts at about $1,900 and has two analog channels, 16 digital channels (for logic analyzer functionality), and up to a 500 MHz bandwidth. It's a great general-purpose lab oscilloscope. It has a big brother, the MDO32, which has up to a 1 GHz bandwidth, as well as spectrum analyzer and signal generator functionality. The MDO32 starts at about $4,600.

Test equipment companies love to try to get you to buy the expensive add-ons: advanced triggering options, protocol decoders, and the like. If you plan to buy these, be aware that they can quickly double the cost of the instrument compared to its base configuration. If you're trying to save money, make sure you insist that you need only the bare options. If you're buying used, extra options can be good or bad. On the one hand, the seller can try to charge you more for options that happen to be on an instrument even if you're not planning to use them. On the other hand, some sellers don't know or care, and you can sometimes get what would have been a really expensive options package for much cheaper.

There are plenty of old scopes on eBay made by HP, Agilent, Tektronix, and Rohde & Schwarz. Don't be afraid of an analog oscilloscope. They may be giant, heavy, and lack the fancy interfaces of digital oscilloscopes, but if they have the specifications you need, they'll work great. Note, though, that analog oscilloscopes typically don't have any storage options, which means you can't use them for certain applications like testing serial buses. If you go the eBay route, make sure to have the device calibrated before you use it. The original manufacturer *may* still offer this service for the undoubtedly discontinued model you bought, but other test equipment distributors or rental companies can calibrate it if the manufacturer won't.

An oscilloscope is only as good as its probes. If you have a scope with a 10 GHz bandwidth but crappy probes, you're not going to see

anywhere near 10 GHz of bandwidth. Unfortunately, high-performance probes can cost up to $10,000. Don't forget to factor this into your budget when buying an oscilloscope.

Logic Analyzers

While many people's first big lab equipment purchase is an oscilloscope, it should probably be a logic analyzer instead. Most of the time when I want to see a time-domain signal, it's from a communication bus like I2C or SPI, and a logic analyzer works much better than an oscilloscope for that. You can see the clock and data lines at the same time, and you can also decode the bus traffic directly. Some oscilloscopes can do this, too, using multiple channels and potentially a software decode option, but you can usually get a logic analyzer for much cheaper.

For most applications, you can get by with a Saleae logic analyzer. This is a little USB device with either 8 or 16 channels that runs on free software from your laptop. The software is really fantastic, and you can download it to play around with before you decide to buy the hardware. The downsides of a Saleae are that it's not great at advanced triggering and that it's not as fast as more expensive analyzers. However, bus decoding works great, and Salaes can even decode low-speed USB. The cheapest model starts at $500.

More expensive logic analyzers have many more channels available, can capture faster events, and can do more complex analysis of signals. That said, fewer manufacturers are making purpose-built logic analyzers and are instead incorporating that functionality into oscilloscopes.

Test Leads

Every lab needs test leads. Use Pomona leads as often as you can afford to. They're more expensive, but they're by far the best test leads out there. You'll need several types of cables: banana jack to banana jack, banana jack to minigrabber, minigrabber to minigrabber, minigrabber to BNC, and banana jack to BNC. Banana jacks plug into your multimeter, power supplies, and low-frequency signal generators. BNC jacks plug into oscilloscopes, frequency counters, and some signal generators.

Minigrabbers attach to your circuit. These connectors work only for DC and low-frequency signals.

For RF signals, you'll need coax cables (probably SMA, depending on the frequency you're using), terminators, directional couplers, and other application-specific adapters. If you're using a VNA, consider getting some instrumentation-quality cables for it. These are rated for high frequency, have very low loss, and maintain a constant phase despite being moved around. Instrumentation cables can cost thousands of dollars apiece. Yet again, eBay is a great place to find used cables for much cheaper. Expensive cables will often come with a sheet of measurements taken on those exact cables so you know what the loss and S11 is. It's a good idea to take these measurements yourself on cheaper cables that don't come with them. Often, having lossy cables doesn't really matter, as long as you know what the loss is and can account for it.

Spectrum Analyzers

If you're going to be working with RF, you're going to need a spectrum analyzer. New spectrum analyzers from Keysight or Rohde & Schwarz usually start at about $20,000 and can run up to over $100,000. Price is most strongly correlated to maximum frequency.

You can get used analog spectrum analyzers for much cheaper than you might expect. The Tektronix 492 has a massive range of 10 kHz to 21 GHz. You can find these on eBay for about $1,000 to $2,000. If you buy one, make sure it has options 1, 2, and 3. However, the interface on the Tek 492 is a little vintage and doesn't include all the buttons and functions you would expect from a modern spectrum analyzer. Instead, one of my favorite models is the HP 8595E. This is a great little instrument that goes up to 6.5 GHz and can be had for about $2,000. The HP 8596E is in the same family and goes up to 12.8 GHz for $3,000 to $4,000.

Handheld spectrum analyzers are nice for working in the field or carrying around the lab. The FieldFox from Keysight (and previous versions from Agilent) is a high-performance instrument that runs off battery power and can even act as a two-port VNA to measure S

parameters. The RF Explorer (*https://rfexplorer.com*) is a low-cost, low-bandwidth, portable analyzer that starts at only a couple hundred dollars. Different models cover different bands, but the highest-frequency model can reach about 6 GHz. It can also plug into your computer and be used with more full-featured software.

USB spectrum analyzers are also available, often priced at a few thousand dollars. Alternatively, you can use a software-defined radio as a USB spectrum analyzer. These are available at lots of different performance levels and price points. The cheapest example is the RTL-SDR, which costs about $30 and can measure up to 1.75 GHz. It has a lot of software support but is limited to a fairly small bandwidth. The other advantage of SDRs is that they can demodulate the signal that they're measuring. This is a feature usually relegated to very expensive modern spectrum analyzers with equally expensive options enabled. However, purpose-built spectrum analyzers will typically have more sensitivity, higher resolution, and a better noise figure than an SDR.

A good accessory to get for your spectrum analyzer is a near-field probe. They typically come in a kit containing E and H field probes as little wands that connect to your spectrum analyzer. Unless you have a really nice spectrum analyzer with a very low noise floor and good input stage, you'll probably need some external preamplifiers for these probes. Sometimes they're available with the kit, and sometimes you may need to buy them separately. These small probe antennas allow you to "feel around" your PCB or device to pick up where stray fields may be coming from. Used correctly, near-field probes can alert you to EMI problems before you take your device to a compliance testing lab.

Vector Network Analyzers

Vector network analyzers are the most expensive piece of test equipment in most labs. New VNAs made by companies like Keysight, Rohde & Schwarz, and Anritsu can easily hit $50,000 to $100,000 on the low end. They also have a plethora of options and upgrades available that a sales engineer can help you navigate. Most VNAs are two-port, meaning they can measure four S parameters. Four-port VNAs are also available, and modular VNAs can provide up to 32 ports to work with.

If you're on any sort of budget, look for old HP, Agilent, Keysight, Rohde & Schwarz, or Anritsu VNAs on eBay. They can be had for surprisingly cheap ($1,000 to $10,000). Note that some old VNAs come in two parts: the display and the test set. Before you get excited that you found a VNA for $500, make sure that it includes both the display and the test set, or that it has the test set built in. Even if you work with very high frequencies, some older VNA models can work up to 110 GHz. My favorite used VNA is the HP 8753D. It's available up to 6 GHz and costs between $3,000 and $5,000.

If you have only hundreds of dollars and need a VNA, you're going to have to sacrifice performance. You can buy USB VNAs (although these can quickly get into the price range of a used VNA from eBay), or you can use a handheld VNA. Both of these options are more affordable, but they won't have the same frequency range and will have a reduced set of features. A really cool recent entry into the market is the NanoVNA (*https://nanovna.com*). This is a handheld, two-port VNA available in several frequency ranges: 50 kHz to 900 MHz, 10 kHz to 1.5 GHz, and 50 kHz to 3 GHz. The most amazing part is the price: $75 to $125. These have become really popular with hobbyists recently, and if you work with anything in those frequency ranges, it's hard to see a reason not to buy one.

Unlike a multimeter or spectrum analyzer, you need to calibrate a VNA every time you turn it on. This is done with a calibration kit that contains a short, an open, and a load. Calibration kits can be shockingly expensive because they work over a very large frequency range; buy them used if possible.

There are two types of calibration kits: mechanical and electronic (or e-cal). Mechanical calibration kits contain a short, an open, and a load in a connector form factor. One by one, you screw these onto the ports of the VNA for calibration. An e-cal kit saves time by allowing you to connect each port of the VNA to the e-cal kit and then plug in the e-cal kit to the VNA with a USB cable. The VNA controls the e-cal device through USB and switches between a short, an open, and a load automatically, rather than requiring you to manually attach and remove each successive calibration connector. Manual calibration kits are

usually cheaper than e-cal kits. You may be able to build your own calibration kit by modifying some connectors, but it's unlikely that your 50 Ω resistor looks like 50 Ω from 0 to 26 GHz. What you can get away with depends on the accuracy you need and your frequency range. For most applications, it's not worth it to skimp on the calibration kit when you spent thousands of dollars on the VNA itself. You want your measurements to be accurate.

Since VNAs are sensitive to phase, you also need to calibrate out any test cables or pigtails you're using. To do this, you can either perform a calibration with the cables connected or use what's called a port extension. On an old VNA, you need to manually tell it the length of the cables you're using and it will compensate for the added phase shift. New VNAs can do this automatically.

Small Hand Tools

Besides all the large items, you'll need a variety of small hand tools to help with your work. This section covers the drills, cutters, screwdrivers, and other tools that should be readily available in a well-stocked lab.

A microdrill (sometimes called a micromotor) will allow you to perform extremely precise and clean rework that would otherwise be impossible. For example, you can use one to excavate small areas of a PCB to perform rework on inner layers. A microdrill looks and sounds like a dental drill, and it basically is one. It's kind of like a smaller and more precise Dremel tool. It has a handpiece and different burr tips, as well as speed control and sometimes a foot pedal to turn it on and off. A microdrill is much faster than an X-Acto knife for cutting traces, but you need to be careful not to cut too far. Good units cost several hundred dollars. You can find cheaper models on eBay, but they don't have a lot of torque.

A good pair of tweezers is absolutely necessary. Jeweler or surgical tweezers are best. Titanium tweezers are strong and flexible, but they can be quite expensive. Avoid tweezers that are magnetic or that can be magnetized, since it can be very frustrating to try to release parts that keep sticking to your tweezers. Having a couple pairs of cheap $5

tweezers around is also good for tasks that risk damaging your more expensive pairs. Get as fine a tip as you can. A good pair of tweezers should be sharp, and the tips should make full contact when squeezed shut. Both curved and straight-tip tweezers are good to have around. Some very large parts may be too wide for normal tweezers to grasp, so some larger pairs can be helpful, too.

Tweezers aren't the only medical tool that finds use in an electronics lab. A good pair of hemostats can be great for reaching difficult places in an assembly and can also be used in lieu of a pair of tweezers for large components. Dental picks are helpful for scraping away solder mask, flux, or other debris on a PCB. A scalpel is also a good upgrade from an X-Acto knife, but make sure you get a #10 blade with it. Just be careful because it will be substantially sharper than an X-Acto blade. Try to get surgical-grade tools if you can; the cheap stuff just isn't worth it.

Diagonal cutters, sometimes called angle cutters or flush cutters, are constantly used. The most common task is cutting excess length from soldered through-hole components, but they can also be used to cut almost anything else. A word of warning though: Don't attempt to cut thick wire with small angle cutters. These aren't designed to bite through larger-gauge wire and will cause dents on the blade that ruin the tool. Use wire cutters to cut anything thicker than about 20 gauge.

Wire strippers come in two flavors: manual and automatic. Have at least one pair of each. Automatic wire strippers are very fast and make a perfect strip each time. Manual wire strippers work with any gauge wire (including those that your automatic stripper can't handle), but there's often a danger of biting down too deep and damaging the wire under the sheath. You can also buy thermal wire strippers, which use heat to melt the insulation around the wire and pull it off.

Needle probes are one of my favorite tools. They're very cheap and I use them constantly. A good pair will be all metal and conductive. You can use a minigrabber cable to grab onto the top of the probe and use a pair of needle probes to test very hard-to-reach places on a PCB with a multimeter. Besides being able to reach tight places, needle probes can also be pressed down into a trace or flux-covered solder joint to make

good contact that a normal multimeter probe might be too blunt to make.

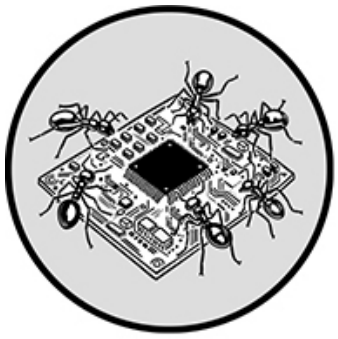
A thermal imager can help locate dead shorts or thermal management problems quickly. Until several years ago, the only thermal imagers available cost thousands of dollars and were very fragile. Today, smartphone thermal imagers like the FLIR One or Seek Thermal let anyone have access to this technology for only a couple hundred bucks. The resolution and frame rate won't be as good as the expensive versions, but they're usually sufficient for simple thermal analysis of a PCB. Another cheap way to get a good thermal camera is to, again, check eBay. You can sometimes find FLIR cameras with a firmware hack that unlocks software-restricted features available only in more expensive models (but that involves the same hardware as the less expensive models). Search *FLIR upgrade* to see if any are available. You can also perform this hack yourself using instructions and files available on various forums. But beware, if anything goes wrong during your hack attempt, or if the hacked device you buy breaks, your warranty is extremely void and you'll need to buy another camera.

You're also going to need a full screwdriver set. IFixit and other companies sell a kit with a small screwdriver and 64 different bits that will cover almost any application you encounter. Another neat tool is a screw-holding screwdriver. A company called Klein Tools makes these, and they have a special bit that can hold the screw to the tip of the screwdriver, even if it's not made of metal or magnetic.

Conclusion

Building a lab is really fun, especially for those who like to geek out about gear. If you have a big budget, the latest equipment has some really amazing capabilities. If you have a small budget, keep scouring eBay and you'll still be able to find accurate, dependable instruments (though you may need to get them calibrated). You'll spend a lot of time in your lab, so make it a comfortable place to work with equipment you trust.

12
FABRICATION AND ASSEMBLY



This chapter examines the process of fabricating and assembling a PCB. If you go into this process with a good understanding of how it works, you'll save tremendous time and effort. Many electrical engineers make the mistake of thinking they can just pass their design to a PCB manufacturer and let them handle everything, but there are lots of hidden options and instructions that you can and should give fabricators to make sure they build what you want them to build. Knowing what's easy or hard to manufacture is also valuable, allowing you to avoid time-consuming or expensive processes if they aren't absolutely necessary. Even just knowing what files to send can be an unexpected challenge if you aren't prepared. We'll go over all of this and more as we discuss ways to get your boards made quickly and smoothly.

In this chapter, you'll often see the terms *fabricator*, *contract manufacturer (CM)*, and *assembler*. While people in the industry sometimes use these terms interchangeably, they have distinct meanings. A *fabricator* is a company that just makes printed circuit boards. An *assembler* is a company that places your components on a circuit board and solders them on. Some fabricators can also act as assemblers. A *CM* is usually a larger company that can do the jobs of a fabricator and an assembler, and can provide testing services, make mechanical parts and enclosures, build your final product, and even warehouse and drop-ship your packaged product. Fabricators and assemblers are often used for prototyping or hobbyist projects, whereas a company will use CMs to produce thousands of a product. We'll get into more detail about fabricators, assemblers, and CMs throughout the chapter.

Preparing for Fabrication

There are an overwhelming number of PCB fabricators and assemblers, varying in cost, speed, and quality. Choosing one can be daunting, but you can get simultaneous quotes from lots of different manufacturers from *https://pcbshopper.com*. This service lets you put in the basic design requirements of your board and returns the cost and lead time for fabricating it from 22 different PCB manufacturers. It will also give you a quote for assembling your board from seven different assemblers. While this can be a great way to find the cheapest manufacturer for your design, it's up to you to vet the quality and capability of whoever you choose.

Once you've selected a fabricator or assembler, be careful not to underestimate the time between when you send out your design files and when they actually start work on the board. This is one of the easiest ways for a project to get unexpectedly behind schedule. If the factory is missing files, can't open them because they're in the wrong format, or finds any issues that they think will impact the fabrication, they'll place a hold on the order and work with you to resolve it. Because of the speed at which modern PCB fabricators move, a single hold order, even if it's resolved within a few hours, can move your ship date back by a day or more.

To avoid holds, run DRC and ERC checks before you generate your output files. All PCB fabricators will have a page on their website listing the DRC requirements for fabricating with them, so you can make sure you're checking against their specific design constraints. Some manufacturers will even provide a downloadable DRC file for common CAD tools so you can simply import it and run it. Other manufacturers have an online tool that allows you to upload your design files and automatically check them for errors.

There are a lot of files that a manufacturer needs to fabricate a PCB. Table 12-1 lists the most important files to send for a two-layer board.

Table 12-1: Two-Layer PCB Fabrication Files

File extension	Description
.gto	Top layer silkscreen
.gtl	Top layer copper
.gtp	Top layer solder paste
.gts	Top layer solder mask
.gbo	Bottom layer silkscreen
.gbl	Bottom layer copper
.gbp	Bottom layer solder paste

File extension	Description
.gbs	Bottom layer solder mask
.gko	Keepout
.drl	Drill file

The file format that every PCB fabricator uses is called the *Gerber*. This name is left over from the company that invented it in the 1980s, but it's ubiquitous in electronics design. The current version is called RS-274X, so make sure your CAD program is generating that format instead of the obsolete RS-274-D. (All modern tools should use the correct version by default.) Since Gerber files are just text files that contain graphics information about where features are on your board, the file extension your CAD program generates may not be exactly the same as what's in Table 12-1. Your CAD program may also generate many more output files than those listed here. For example, if you have more than two layers, you'll need another file for each layer. There may also be an outline file, mechanical layers, a netlist (usually in IPC-D-356 format), or other information about the design. And don't forget the drill file! It's really easy to forget to include it, since some CAD programs export the drill file separately. The standard drill file format is called NC Drill, which is different than the Gerber format.

Even if the PCB fabricator doesn't ask for it, send them supporting files to help them better understand the design. One example of this is a README file. It's good practice to include a *readme.txt* file that lists the files you submitted, describes what they are, lists the stackup you're fabricating on, and provides any other design-specific information that they might need to know. Sometimes the fabricator will place a hold on a design if they see something unusual or contrary to what most designs do. You can try to head this off in your README by calling out what might look like a mistake but actually isn't. An alternative to sending a README file with specific instructions is to put those instructions as their own layer in your Gerbers. There's less of a chance that your fabricator will miss the information this way because they're going to have to look at the Gerbers to fabricate your design.

If you're using a custom stackup, be sure to include a diagram that shows the material, substrate thickness, copper thickness, and relative permittivity for each layer. If you have any impedance-controlled lines, you should have a list of the nets that need to be impedance controlled, their desired width and impedance, and which layer they're referenced to.

There's a lot of information that you can call out in fabrication notes, and the more specific you are, the closer your board will be to what you expect. Appendix C

shows an example set of fabrication notes for a design that you can use as a template for fabrication notes on your own design.

It's a good idea to take one last look at your generated Gerber files before you send them out to make sure that everything exported the way you think it did. Two good programs to use for this are gerbv and ViewMate; gerbv is free and open source, and there's a free version of ViewMate as well. Using software like this is also good for discovering missing files, since you'll notice missing information when you open all the Gerbers at once and see them stacked on top of each other. You can find links to gerbv and View-Mate on this book's website.

Beyond carefully vetting your Gerbers, another way to keep the manufacturing process on track is to provide a properly prepared BOM containing all the important information needed to purchase the right components from the right suppliers. This can speed the procurement process considerably. A BOM can have lots of different fields, but the most important ones are the reference designator, the quantity, and the manufacturer part number. Other useful pieces of information to provide are the distributor part number, the unit cost, the subtotal cost, a description of the part, the part's footprint, and a link to the part on the distributor's website.

Buying Parts

You have two sources for buying parts: a distributor or the manufacturer. For the majority of your parts, and for smaller-volume builds, you'll likely source all of your parts from distributors. Common, reputable distributors are DigiKey, Mouser, Newark/Farnell, and Arrow. For RF parts, RFMW and Richardson RFPD are good places to go.

One interesting and unadvertised feature of the Mouser website is that the search bar understands DigiKey part numbers. This is useful when trying to find parts that may be out of stock on DigiKey or if you're sourcing parts for a BOM on Mouser and have only DigiKey part numbers. Sadly, this feature doesn't go both ways: The DigiKey website doesn't understand Mouser part numbers.

Many distributor websites have a BOM import tool. It typically lets you upload a BOM as an XLS or CSV file and automatically searches and adds all of the parts to your cart at once. The tool also usually alerts you to any parts that are out of stock or have insufficient stock, and sometimes it even suggests replacements. It's a good exercise to try to add all of your parts to your cart before you're ready to order them, just to make sure you'll be able to actually buy everything. The service mentioned in Chapter 9 for checking multiple distributors at once, Octopart.com, also has a BOM upload tool. It can tell you who has how much stock of each part and if you're in danger of using unavailable parts.

When you're exporting your BOM to upload to a distributor website, make sure to use the original manufacturer part numbers rather than distributor part numbers to find parts. This will make your BOM distributor-agnostic and will keep you from having to update your part numbers if a distributor changes anything. At a minimum, BOM upload tools require you to include columns for the manufacturer, part number, and quantity.

When ordering parts, some distributors allow you to provide a custom field for each part, sometimes called the *comment*, that they'll print on the bag of parts they deliver. I've found that it's very helpful to use this for indicating the reference designator for each component, the name of the project, or both. This helps you stay organized, especially if you have multiple projects going at the same time. If your PCB has silkscreen labels for each reference designator, it's also much faster to hand assemble your board because you can match each bag with the silkscreen label on the board. If you have an in-house inventory system with custom part numbers, you can also put those part numbers in the comment field.

If you go to buy a part and notice that the price has suddenly increased to an incredible level, it's likely that either the distributor has a small number of those parts left and isn't planning on buying more or the part has become end of life. In either case, the price spike is to discourage you from using that part. You should take the hint and find a replacement; it's very likely that there's some kind of ongoing supply chain issue and you won't be able to reliably get the part.

Some distributors offer to put parts on a reel for you, even if you buy less than a full reel's worth of parts. DigiKey calls this service their Digi-Reel, for example. This is great if you're using a pick-and-place machine, but it can be annoying if you're assembling by hand. Reels take up a lot of space and make it very difficult to store a small number of parts if you have any left over. A small antistatic bag with a label on it is much easier to accommodate than a 6-inch-diameter reel with a short piece of cut tape that's always falling off. Only get small quantities of parts reeled if you actually need them on a reel. It'll also be cheaper since DigiKey charges a fee of several dollars per Digi-Reel.

Make sure you order parts far enough ahead of your build that there's no bottleneck waiting for parts to arrive. If you're using a pick-and-place machine or an assembler, you'll need to get parts before you plan to start assembly. Using a pick-and-place machine requires setting up the feeders with the parts and programming the machine. The earlier you get parts to your production team, the more likely you are to stay on schedule.

Some component manufacturers and turnkey PCB manufacturers will program parts for you before or after production. For example, Microchip (*https://www.microchipdirect.com*) has a service that allows you to buy microcontrollers in tape and reel preprogrammed from the factory with your

firmware image. CircuitHub.com allows you to upload your firmware image when you're ordering your boards and have it programmed during manufacturing. This can be a great resource if you're building a large number of boards, since you don't need to build a programming jig or worry about the logistics of flashing every single board. The downside of using a preprogrammed chip, however, is that you need to have a completely finished firmware image to give to the manufacturer. If you ever need to update anything, you'll have to connect to every single board and flash them manually, essentially defeating the purpose of having the manufacturer do it in the first place. Depending on your design, and if it's capable of connecting to any networks, you may be able to have the manufacturer upload a version of your firmware that does automatic updates by checking for a new firmware image at a hardcoded address or performing over-the-air updates. Again, though, if anything goes wrong with the automatic update, you have to flash each board one by one.

Finding and Working with Contract Manufacturers

If you're building a handful of PCBs for prototyping, you can use a PCB fabricator that will just make, and maybe assemble, your boards and ship them to you; however, if you're producing a full product in large quantities, you'll need to engage a CM that will not only make and assemble the PCBs but also produce mechanical parts and enclosures, perform final assembly, test your product, pack it, warehouse it, and ship it. Choosing a CM is a big deal. If you choose poorly, your product can easily end up delayed or badly made, and switching CMs is a long, expensive process. The best way to find a CM is to get a warm introduction from someone who's happy with their CM. Talk to your friends, people you know in the industry, people you know at other companies, your investors, your investors' other portfolio companies, and so on. You can also cold-contact CMs by finding them on sites like Global Sources, HKTDC, Maker's Row, or even Alibaba.

Don't assume that it will be cheaper to manufacture in China. While this almost always used to be the case, rising wages in China have changed the economics of manufacturing. If you're in the US, choosing a CM that's also in the US will make travel to the factory much easier and may reduce the cost of shipping and logistics for the finished product (since it doesn't require shipping across the Pacific Ocean). Intellectual property laws in the US are also much stronger and are actually enforced, so if you're worried about IP theft, manufacturing in the US is a good way to mitigate that risk. Mexico can be a good middle ground between the US and China; it can be more cost effective while still being fairly easy to travel to.

Other advantages of manufacturing in North America are that the time zone differences between you and your factory will be smaller and that you'll encounter

fewer language and cultural barriers with your factory. Miscommunications between you and the CM are a major contributor to manufacturing errors and delays, and you're especially vulnerable to miscommunication if you and your CM are several time zones apart, don't share a first language, or are from two different cultures. Whatever CM you use, it's imperative that you communicate clearly and frequently, probably more clearly and frequently than you think is necessary.

Once you start talking about money with your CM, they'll probably want either net 30, net 60, or net 90 terms. This is standard lingo that just means you need to pay in full either 30, 60, or 90 days after delivery. There may be other financing options available specific for your situation and what your CM is comfortable with, but the important thing to know is that you need to talk about payment and figure out a solution that works for everyone.

Whichever CM you pick needs to be closely aligned to what you're trying to make. Obviously, you wouldn't go to a textile factory to produce a phone, but you also probably don't want to go to a factory that makes only TV remote controls. Find a CM that's already making something similar to your product. They'll have experience with similar components and requirements, which will ultimately save you both time and money.

One of the first questions your CM will ask you is how many units you want them to make. Factories come in many different sizes, and engaging the wrong size factory will result in either waste or stress. Do your best to come up with a realistic estimate of the number of units you'll want produced for up to two years of production. The larger your error on this number, the more difficult it will be to keep your factory happy. This is especially true if you overestimate the number of units you want, since the CM may hire more people or buy expensive equipment in anticipation of handling a larger order volume, only to be left footing the bill.

Be responsive to your CM; it's in both of your best interests to keep production moving at all times. This is harder if you're based in the US and they're based in China, since they're starting up production for the day just as you're getting ready for bed. To mitigate this, it can be extremely beneficial to have someone at the factory during the beginning stages of production to help fix the inevitable issues that will arise. The CM can fix problems with their equipment, but they won't know what to do if your devices start to fail tests at a high rate. At that point they'll have to hold production until you can get someone into the factory to debug the problem. If you can't have someone in the factory, have your phone on you at all times, read your emails and respond to them as quickly as possible, and be ready to go to the factory at a moment's notice. Nothing is more frustrating than trying to work with someone else and having them be unresponsive.

Ideally, you won't have to make any design changes after production has started, but if changes are needed, make them as minimal as possible. First, try using the

contingency features that you designed into your PCB. This will be the least expensive approach, since all you'll need to do is change what parts are populated where. This isn't always possible, though, and if a significant change is necessary, it may end up being quite expensive and will delay production. As you may expect, the further along you are in production, the more expensive changes are.

Respecting Your Contract Manufacturer

Don't assume you know more about manufacturing than your CM. That may sound like an absurd assumption, but it's a common problem: According to Fictiv's 2018 hardware review survey, 54 percent of developers consider themselves very or extremely knowledgeable about manufacturing processes, but only 27 percent of manufacturers consider their clients to be very or extremely knowledgeable about manufacturing. Because of this imbalance, people submit things that are unmanufacturable to manufacturers all the time.

You need people with manufacturing experience to review your design. If you don't have anyone on your team who has that experience, you can contract someone, or even better, work with and learn from your CM. They've probably been producing electronics longer than you have, and they can offer suggestions to improve your DFM. They'll have a lot of intuition about the manufacturing process that can help keep your product on schedule and free of problems.

Respect the CM's team and treat them as your partners. If they ask you to change something, it's because they want to give your product the best shot possible at being manufactured correctly on the first try. The most common place you and your CM will butt heads about this is on industrial design. Big companies like Apple famously go to extreme lengths and cost to never sacrifice any industrial design requirements. But you aren't Apple (probably), and if you decide to act like Apple when you aren't, your product will never launch. You'll run into preventable manufacturing problems because you don't have the time, money, or expertise to refuse to compromise on design.

Navigating Standards and Certifications

It's important that your fabricator use a standard and proven process to inspect and control the quality of your boards. There are three main standards that your fabricator probably uses. The first is DOD-STD-100, a US military standard for producing mechanical and engineering drawings for all kinds of purposes, not just PCB design. The concepts in this standard are so widely known and followed that no one should have to refer to the standard to figure out what your drawing is trying to show. That said, explicitly calling out this standard removes any ambiguity for the person interpreting the drawings. Consulting the standard's best practices

for clear engineering drawings will also be helpful for *you* if you don't have much experience creating such drawings.

The second main standards your fabricator will use are IPC-6011 and IPC-6012 for rigid PCBs or IPC-6011 and IPC-6013 for flex circuits. These standards specify the level of inspection and testing done on a board. There are four tiers, from least to greatest stringency:

Class 1 Ensures the lowest level of performance, with minimal inspection and testing. This is the cheapest option. IPC describes Class 1 as general electronic consumer products, including games, some computers and peripherals, and even some military equipment, where functionality is less important.

Class 2 The normal level of inspection and testing; most boards are manufactured at this level. If you don't specify an IPC class, this is the class your fabricator will use. IPC describes Class 2 as dedicated-service electronic products, like communications equipment, sophisticated business machines, instruments, and military equipment, where high performance and extended life are required and for which uninterrupted service is desired but not critical.

Class 3 For high-reliability applications like safety- or life-critical devices where a failure may put lives in danger. Military and aerospace devices are also usually manufactured at this level. IPC describes Class 3 as high-reliability electronic products, like equipment for commercial and military use where continued performance or performance on demand is critical. Equipment downtime cannot be tolerated and full functionality is required at all times, such as with life support and critical military systems.

Class 3/A Even more stringent than Class 3 and specifically designed for space and military avionics.

As you might expect, the higher the class you specify, the more expensive the process will be. However, the difference between a Class 2 board and a Class 3 board isn't as great as you might think. It's not uncommon that a Class 2 board may happen to meet most or even all of the requirements of Class 3.

The final standards that your fabricator will use are IPC-A-600, which defines the "acceptability of printed boards," and IPC-A-610, which defines "acceptability of printed board assemblies." IPC-A-600 goes hand in hand with IPC-6012 and IPC-6013 and pertains specifically to inspection of a finished board. It helps your fabricator know when to accept or reject a board because of manufacturing defects. It's a good idea to read these standards so you can better design your board to minimize scrap.

Certifications can be one way to ascertain the quality and abilities of a PCB manufacturer or assembler. Here's a list of common certifications:

ISO 9001 A comprehensive quality-management certification. Good manufacturers will have this. There are multiple versions, one from 2008 and an updated version from 2015.

AS9100C/D The same thing as ISO 9001 but with extra requirements for aerospace applications. The last letter in the name is the revision. The current revision as of 2025 is revision D. At some point in the future, this will be replaced with IA9100, which will include ISO 9001.

ISO 13485 A quality-management system for the design and manufacture of medical devices. This is a standard, and manufacturers can be certified as being compliant with it. The FDA may require this certification if you're making a medical device.

International Traffic in Arms Regulations (ITAR) Indicates that the company has been registered with the State Department's Directorate of Defense Trade Controls. While not a certification, this designation is necessary if the company will be exporting physical things or information related to defense that's covered under ITAR. Products regulated under ITAR have a lot of other requirements as well, but that's outside the scope of this book.

UL PCB Flammability certification. There are two versions: flame only and full recognition. Flame only just ensures that the materials and processes used will result in a board that meets certain fire ratings. Full recognition also checks tolerances and bounds on the final product (things like solder mask thickness, copper thickness, and so on). Good manufacturers will have at least one of these certifications.

Dodd-Frank Conflict Minerals Statement Not a real certification, but important. Gold, tin, tantalum, and tungsten are conflict minerals, meaning that buying them from a disreputable source can result in you indirectly financing conflict in the Democratic Republic of the Congo, where they're mined. Having this statement means that you've procured these materials through State Department–approved channels that don't support this conflict.

REACH A European directive that bans certain hazardous chemicals from any part of a product (including the electronics, paint, enclosure, and so on). This isn't a standard, but a company can say they're REACH compliant.

RoHS The same thing as REACH, but it applies only to the electrical part of a product (the components, solder, and PCB).

MIL-PRF-31032 A military standard for making PCBs. If someone says a board is MIL-SPEC, they probably mean that it meets this standard. This is also not a certification; it's just something you can be compliant with.

MIL-PRF-55110 An old version of the military PCB specification, now superseded by MIL-PRF-31032.

IPC A huge body of standards relating to PCB production and assembly. Good assemblers (the individual people, not the company) are IPC certified. Since IPC also makes standards, it's possible to be IPC compliant but not IPC certified.

JEDEC Another large set of standards that guide PCB production and assembly. Good manufacturers should be compliant.

Beware that just having these certifications doesn't automatically mean your boards will be perfect. Some less scrupulous companies let their certifications expire but still claim to have them, and of course it's possible that the company was once able to pass these certifications but no longer can (even if the certification hasn't expired). You should also make sure the certifications the company claims to have actually exist. Some Chinese fabricators have been known to put certificates and plaques up on their wall that look real but aren't from any official standards organization and are therefore meaningless.

PCB Fabrication

You have three options for PCB fabrication: Etch it yourself, mill it yourself, or have a factory do it. Given the cost and abilities of modern PCB fabrication companies, you should almost always have a factory do it, but we'll discuss all three options here.

Etching

Etching a PCB yourself is best thought of as an educational exercise. It's good to do once, but it's very limiting, requires caustic chemicals, and isn't a realistic option for most real-world boards. You should really only do it if you need a simple PCB within a few hours. Figure 12-1 shows an example of a homemade etched PCB.

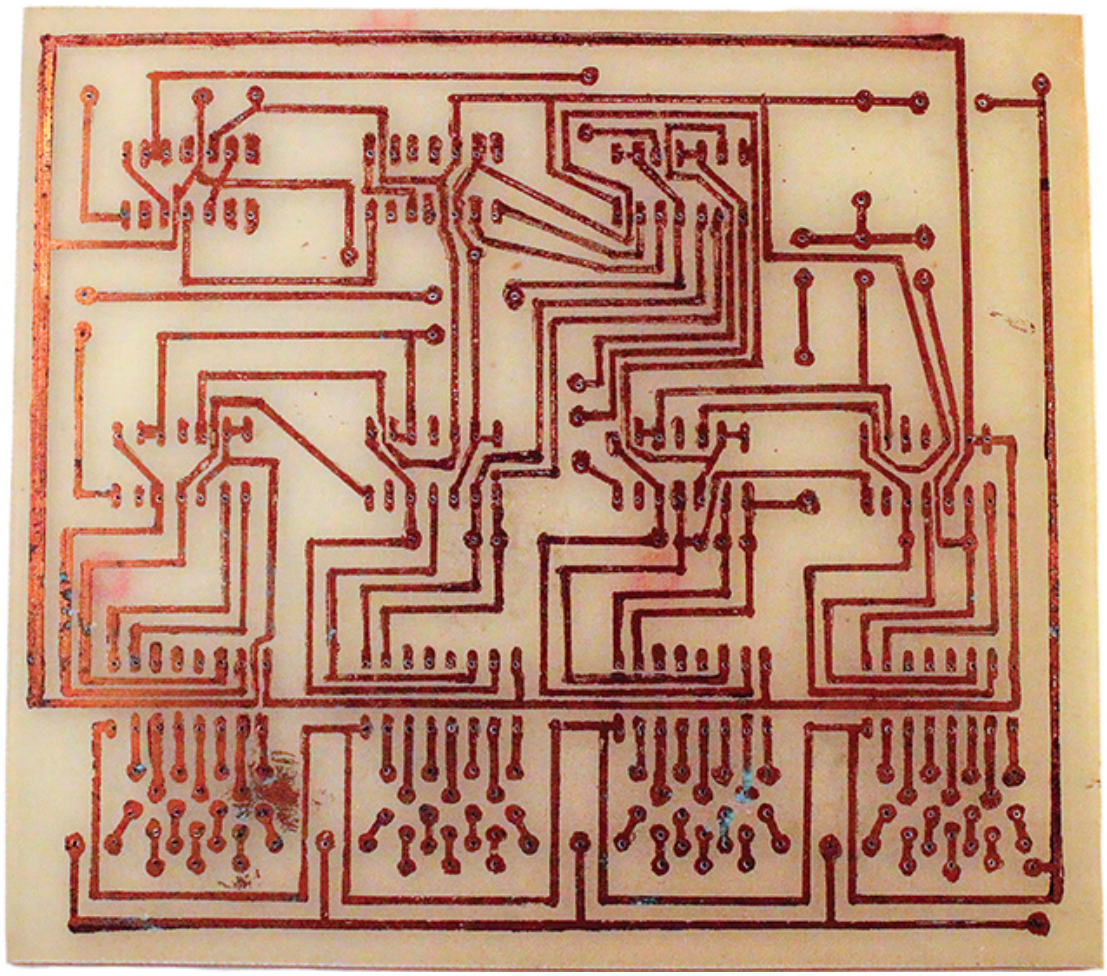


Figure 12-1: A homemade PCB

If you're etching your own PCB, you can typically have copper only on a single side, and small features (less than about 10 mils) aren't possible. You'll have to manually drill through holes, and they won't be plated unless you stuff a wire through the hole and solder it to both sides or use very tiny rivets. Worse, it's also easy to accidentally over- or under-etch your board, resulting in defects like those shown in Figure 12-2.

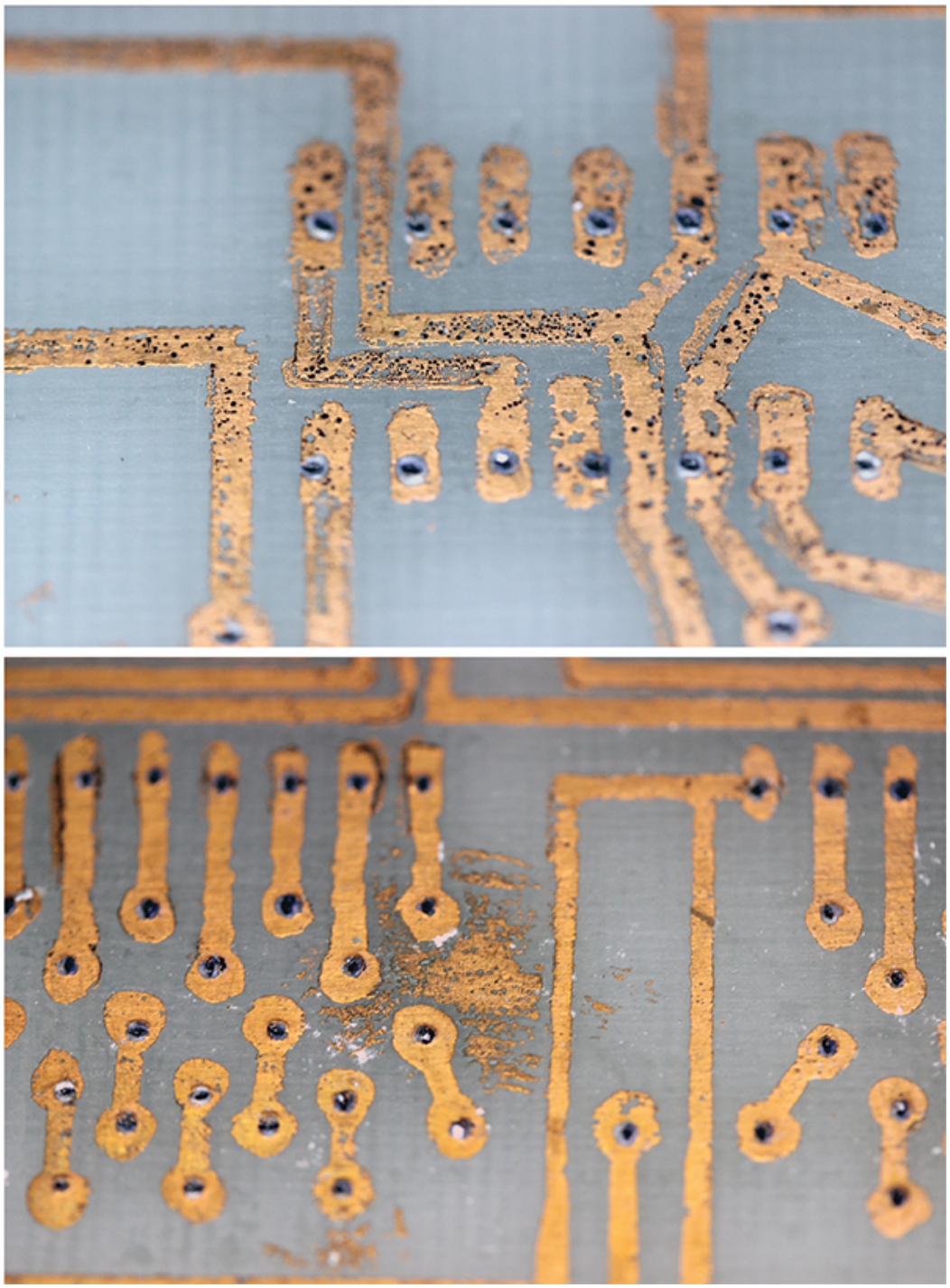


Figure 12-2: Two details from the PCB in Figure 12-1, showing over-etching (top) and under-etching (bottom)

In the top example, the copper trace has either been poorly coated in photoresist, left in the etch bath too long, or both. This trace won't be able to carry the current that a trace of its width is normally rated to carry. It's also at risk of causing an open circuit. In the bottom example, extra copper is visible because too

much photoresist was used and some contaminated an area that was meant to be removed. This effect can also be caused by too little or incomplete etching acid coverage. Either way, the defect risks causing a short.

Milling

Another option for manufacturing circuit boards is to mill them yourself. PCB mills work by using an end mill to remove copper from a piece of copperclad board to create the footprints and traces on your PCB. As discussed in Chapter 1, the nice part about this is that you can design a board and have it in your hand in a matter of hours rather than days or weeks. This is very powerful if you're trying to move quickly. However, milling isn't without trade-offs. It involves a large initial equipment purchase as well as expensive consumables (end mills and substrate). Milled boards also don't have solder mask, don't have plated vias or holes (unless you apply a specific process to plate your holes), and can only accommodate features down to about 10 mils. This means that you can't implement any chip with a pitch smaller than 10 mils (which you'll mostly encounter with SMT ICs) on a mill. Instead, mills are best used for through-hole parts, large-pitch SMT parts, and structures like antennas.

There are two main companies that make circuit mills: Bantam Tools and LPKF. The Bantam Tools Desktop PCB Milling Machine (previously known as the Othermill) is a lower-cost option that starts at about $4,000. As the name implies, it's a small machine; it can mill only material that's 4×6 inches.

The second option is one of many machines made by LPKF. (Colloquially, people often refer to such a machine as *an LPKF*, even though that's really the name of the company, not the machine.) They make machines that support both mechanical milling and laser milling. They're much more capable than the Bantam Tools machine. They can create PCBs with spacing as small as 2 mils, and with additional equipment they can make conductive vias, plated vias, multilayer boards, and even add solder mask. On the other hand, LPKF mills are significantly more expensive than Bantam's. The mills without lasers cost between $20,000 and $30,000.

Hiring a Fabricator

If you take the more sane path of hiring a fabrication company, choosing the company is an important decision. Not all fabricators are created equal, and who you choose may change depending on the time, budget, and sensitivity of your design. The fabricators in the following list cost less but won't do any exotic or highly sensitive manufacturing. If you need a simple two-or four-layer board *without* blind vias, particular copper weights, very small trace widths, or other special requirements, one of these companies may be a good option. Many

hobbyists use these manufacturers for side projects, but they can also be used for quick and simple prototypes or proofs of concept.

OSH Park Very popular with hobbyists; good quality. Offers two-, four-, and six-layer services, as well as flex. Normal lead time is 9 to 12 days; fastest lead time is 4 to 5 days. Its four-layer service is on FR408, which is good enough for some RF applications. Pricing starts at three copies for $5 per square inch. OSH Park isn't actually a fabricator itself, but it panelizes designs and handles production through another fabricator. All US based.

Seeed Studio Fusion Offers one-, two-, four-, and six-layer boards on FR-4, as well as flex. Allows for a little more customization than OSH Park. Lead time is 3 to 4 days. China based.

AdvancedPCB Offers a barebones service limited to two-layer designs with no silkscreen or solder mask and a maximum size, but with a 24-hour turn time. They also offer a "33 each" service that charges $99 for three copies of a two-layer board with a 3-day turn. Their "66 each" service charges $264 for four copies of a four-layer board with a 5-day turn time. All of these services are on FR-4. US based.

JLCPCB Produces one-, two-, four-, and six-layer boards on FR-4, with a 2-day turn time, or, for a little more money, a 1-day turn time. A little more customization is possible, similar to Seeed Studio Fusion. China based.

PCBWay Lots of customization possible. Their cheapest service is $5 for 10 copies of a two-layer board with a 24-hour turn time, but they can handle designs with up to 14 layers as well as flex. China based.

If your board has more specialized requirements, such as blind vias, narrow traces, and the like, the following high-performance fabricators are probably a better fit:

Summit Interconnect The largest private PCB manufacturer in North America. It owns some smaller shops that you can use for prototyping, but it's also capable of very high-volume production and complex processes. US based.

Sigmatron Another large international contract manufacturer. Not where you want to go for hobby or small, quick-turn projects, but it's a good place to start if you're shopping around for a CM.

Eurocircuits Popular European fabricator with price tiers for standard, controlled-impedance, RF, and flex boards.

Gorilla Circuits Handles PCB fabrication and assembly. Very high quality. US based.

There are many hundreds of these companies, so keep in mind that these lists represent only a small fraction of what's out there. They're also sorted based on my personal experience with them as a designer in Northern California. The quality of the builds that these companies produce may vary, and a board house that one person had a terrible experience with may have been just fine for someone else. I'm not guaranteeing anything about them. Use these suggestions as a starting point, not as an absolute ranking. The best way to find good companies that aren't listed here is to ask people you know who have fabricated PCBs before. Who did they use? What was their experience like?

How Assembly Works

An *assembler* can be a separate company that receives PCBs after they've been fabricated, or it can be integrated with the PCB manufacturer as a turnkey solution. Assembly is as much an art as a science, so experience is what you should be looking for when choosing an assembler.

There are several ways to approach PCB assembly, depending on the resources available. For a large company doing a large production run of a single product, for example, there will be component engineers, process engineers, and production engineers all working together with one or more external factories and internal production lines to tune the production precisely for that product. In this situation, you'll build custom fixtures to aid in assembly and work to optimize the yield of your assembly line. Assembly of the electronics will be followed by a functional test and then packaging and shipping. At this scale, there are entire companies that are capable of handling each of those steps for you. Because very large production runs are so specialized, we'll focus mostly on smaller production sizes.

Small startups will usually not have the resources to do manufacturing and assembly in house, so they'll contract it out to a factory or turnkey service. Several employees will work with the factory to ensure assembly is done correctly. You'll start by selecting a manufacturing partner. Get quotes from multiple companies that you trust. The best way to start identifying good companies is to look at factories that people you trust have recommended. Talk to other hardware startups (cold email them if necessary) and get an idea of who will do a good job. Make sure they're at least ISO 9001 certified (and that their certification is current), but you may also require other certifications depending on what you're manufacturing.

You can consign parts to assemblers (meaning you ship them components you want them to use), they can procure them all for you, or you can do a combination of the two. Many assemblers charge a markup of about 10 percent on materials and parts they buy, so consign parts if you're trying to save money. You can also sometimes negotiate this markup down.

If you're doing a one-off prototype run, find a company that knows what they're doing and has a lot of experience. If you're doing entire product production runs rather than just PCB assembly, it's more important to establish a relationship with the factory owner. As a small startup, you may not initially be bringing in the same level of revenue as a larger, more established company, so the factory is making a bet on you and hoping that you'll grow and become a long-term partner with them.

If you're a hobbyist or an early-stage startup, using a factory for assembly might cost too much money. However, with the right techniques, you can assemble quality PCBs at home in low volumes. The next section includes some assembly techniques you can use if you're soldering together boards yourself.

Soldering

The best way to learn soldering is to watch videos of good soldering techniques and then try it yourself. (You can find some of these videos on this book's website.) Soldering is a very dynamic process, and it's hard to convey what it should look like as the solder gets to the right temperature and how it flows around component leads without using video. That said, this section attempts to lay out some useful techniques and tips for soldering.

I'm assuming that you're capable of basic through-hole soldering; if you don't know the basics yet, take some time to learn them and practice. I'll instead focus primarily on SMT and hot-air soldering. These techniques can seem intimidating, but they aren't that difficult with a little practice, and they vastly increase the amount and kinds of rework that you can perform, which can save you time, money, and revisions. If you don't want to fabricate your own PCB to practice SMT and hot-air soldering on, there are links to some good practice kits on this book's website. Also, note that almost all soldering is significantly easier under a microscope; I highly recommend buying a good binocular microscope with a ring light.

Soldering-Iron Techniques

Soldering with an iron is all about having the right soldering tip and temperature and using the right solder and flux. If you're having trouble getting a solder joint to flow correctly, either increase the size of the tip you're using, increase the temperature of the iron, or try adding a little flux. Larger tips will heat components faster, although they're harder to get into tight places.

If your soldering iron has an adjustable temperature, make sure you're managing the temperature correctly. If it's too cold, the solder won't melt well. If it's too hot, you can damage parts or cause traces to peel up from the board. One

mistake that people often make while soldering is applying too much pressure to the iron. Pressing down hard doesn't make the heat transfer any faster and will just damage your soldering tip.

Always apply solder to the solder *joint* and not the tip of the soldering iron. You can *tin* the tip of the iron with a little solder before you start to remove any oxides and improve heat transfer, but when you're actually soldering a component, use the iron to heat the solder joint and apply solder to that joint. You'll know when a solder joint is successful when it looks shiny and the solder "flows" around the joint. If the solder looks dull or beads up, you probably have a cold solder joint, which won't be electrically or mechanically robust. Don't use too much solder, either. You want a nice fillet or chamfer on the pins of your component. Beading up means you've used too much. All you need is enough to cover the pad and pin in a thin layer.

If you're having trouble preventing the solder from balling up on you as you heat it with your soldering iron, make sure your tip is in good condition. Use brass turnings instead of a wet sponge to clean the tip. The brass will do a better job at scraping off the oxide layer and will put less thermal shock on the tip. You can also buy *tip cleaner*, which is a little tray of material that you can use to coat your soldering iron tip before you turn it off. Coating the tip in solder before powering it off will help keep oxide from building up on the tip surface. This oxide is chiefly responsible for the areas of the tip that the solder doesn't seem to stick to.

When soldering SMT parts with an iron, you can use a *raking* technique (sometimes called *drag soldering*) with QFP-type packages to quickly solder every pin: Put down a thin layer of flux across the pins on one side of the package, then place a blob of solder on one side and drag it across the pins with the iron. You should see the flux smoke and the solder grab onto each pin perfectly. If you have any shorts, remove them with solder braid. This technique works best on QFN/DFN packages. The footprint needs to be designed to extend the pad lengths slightly so there's room for a soldering iron tip.

For parts with an exposed thermal tab (like a TO-220), use a large chisel tip to get enough heat for the solder to flow. For really large parts, or parts that are very well grounded to a large ground plane (and thus have a huge thermal mass), use both hot air and a chisel tip to heat the area. This can damage the board or nearby parts, and it can also damage your soldering iron tips (or reduce their lifetime), so be very careful when using this technique. Heat-sink any sensitive parts that you're not trying to work on.

Flux

Flux is a wonderful tool that can make soldering significantly easier. Flux removes metal oxides and increases solder surface tension. If you're having a hard time

getting solder to flow, try adding a little flux and it will usually start behaving. Don't mind the puff of smoke—that's normal. In fact, the smoke you see when soldering is actually flux, not lead.

When you buy flux, make sure you're buying flux for electronics. Plumbing flux isn't the same thing, but the packaging can look similar. Flux is available as a liquid or as a thicker paste, sometimes called tacky flux. Pure-liquid flux can get messy and spread out very thin, but it's good for applying very small amounts and for reworking very small components. Flux paste can be moved around more easily and targeted to a specific area. Flux is usually applied with a syringe or a cotton swab. It's also available in pens. Flux pens look kind of like felt-tip pens, except instead of ink, the tip is soaked in flux. A little bit of flux goes a long way, so using flux pens is a convenient way of dispensing the right amount of flux and having a portable supply of flux with you that doesn't get messy.

There are three main kinds of flux: rosin, no-clean, and aqueous. *Rosin flux* is derived from tree sap (specifically pine tar resin). The most basic form of rosin flux is known as *type R*. You can also buy rosin flux that has chemicals added to improve its ability to dissolve heavier oxide films, which is important when working with lead-free solder since you're working at higher temperatures. Rosin flux that has these additives is called either *type RMA* (rosin mildly activated) or *type RA* (rosin activated). You must clean all three types of rosin flux off the PCB after assembly since it can cause corrosion and other reliability problems if you don't.

The second kind of flux, *no-clean flux*, is made with either natural rosin or synthetic resins. It's essentially a more dilute version of regular rosin flux, and it can still contain some activator chemicals to help remove oxides. No-clean flux does *not* mean that the flux will leave no residue; every flux leaves residue. No-clean flux is simply designed to leave residue that isn't harmful to the longevity of your PCB. Even if you use no-clean flux, you may still need to clean it if you plan to use conformal coating so that the coating evenly adheres to the entire board. No-clean flux is also corrosive before it's been activated by heat, so especially when reworking with no-clean flux, it's important to make sure you have heated all applied flux to activate it. You can bake the entire board to ensure you have activated all flux.

Aqueous (or *water-soluble*) fluxes can be washed and removed with just water and are excellent for soldering. However, they contain very aggressive chemicals, so you must wash them extremely thoroughly. Even a small amount of remaining water-soluble flux can cause device failures.

An IPC standard called IPC-J-STD-004 is used to assign a four-character code to fluxes that describes what they're made of, their activity (how aggressive they are), and whether they contain halides or not. Table 12-2 shows how these codes

work; note that halides increase the ability of flux to remove oxides, but they need to be thoroughly cleaned to prevent reliability issues.

Table 12-2: Flux Composition and Designators

Flux composition	Flux/residue activity	Contains halides	No halides
Rosin (RO)	Low	ROL1	ROL0
	Moderate	ROM1	ROM0
	High	ROH1	ROH0
Resin (RE)	Low	REL1	REL0
	Moderate	REM1	REM0
	High	REH1	REH0
Organic (OR)	Low	ORL1	ORL0
	Moderate	ORM1	ORM0
	High	ORH1	ORH0
Inorganic (IN)	Low	INL1	INL0
	Moderate	INM1	INM0
	High	INH1	INH0

No-clean is the most commonly used flux in PCB assembly today. Assemblers like it because they don't need to invest in lots of cleaning equipment that costs money and takes up valuable floor space. Successful use of no-clean flux requires some experience, but all major PCB assembler companies will have this. ROL0 or ROL1 are great options for no-clean flux. Kester and ChipQuik are both reputable flux manufacturers.

Some electronic components can't survive the flux cleaning process, and the datasheet will note that they are "no-clean" parts. Always check the datasheets for every part that you're assembling, and talk to your assembler about which flux is best for your assembly.

Solder Alloys

You'll need to specify whether your design is leaded or lead-free. Almost all products are now RoHS compliant, meaning they don't contain any lead or other environmentally harmful chemicals (although there are some components that are RoHS compliant by exemption and still do contain environmentally harmful

chemicals). It's pretty easy to design an entire board using only lead-free components without even realizing it. Still, you should check the status of all your parts since it affects the temperature of your assembly process. Lead-free assembly has to use higher temperatures during reflow because of the higher melting point of the lead-free alloy. Leaded components, by contrast, are designed to survive the lower temperatures of a leaded manufacturing process.

There used to be some concern around lead-free soldering causing tin whiskers, but modern lead-free solder alloys don't have this problem. The only real reason you should consider using a leaded process is that it's a little easier to rework. Because of lead's lower melting point, repairing and reworking leaded boards is easier, and you're less likely to damage the PCB substrate itself—for example, by going beyond its glass transition point. When assembling a few prototypes by hand, use leaded solder paste. As long as your design is otherwise lead-free (which it should be), you can easily switch to a lead-free assembly process when you move to larger-scale production in a factory.

There are a huge number of solder alloys available. The most common leaded solder to use is 63 percent tin and 37 percent lead. This is known as a *eutectic* mix because this exact combination will melt together at a lower temperature than any individual metal in the alloy. Other proportions of these two metals will result in a solder that has a higher melting point with one component that's liquid and one that's solid, resulting in a gummy consistency since the material is in a plastic region. This will increase the chances of a cold soldering joint.

One of the most common lead-free solders is SAC305, which is made up of 96.5 percent tin, 3.0 percent silver, and 0.5 percent copper. Silver is an expensive metal, so there are alternative alloys that don't contain silver. A great example of this is SN100C, which is mostly tin, a little copper, and very small amounts of nickel and germanium. SN100C is actually a trademark of Nihon Superior, the company that invented it. The patents have expired now, so there are multiple variants that are available from different companies that use slightly different metal ratios. Some reports claim that SN100C and related mixes outperform SAC305 and even the 63/37 tin/lead alloy.

You can find low-temperature solder pastes (also lead-free) that contain primarily bismuth and have melting points as low as 140° C. Some sensitive components can't survive the high reflow temperatures of lead-free solder or even leaded solder. A low-temperature alloy commonly available is 57 percent bismuth, 42 percent tin, and 1 percent silver.

Solder Paste

You need to put some thought into which solder paste to use, especially if you're doing production assembly. There's an entire field devoted to understanding how

solder paste and other materials flow and behave, called *rheology*. If you hear someone referring to the rheology of the solder paste, that's what they're talking about.

Solder paste consists of tiny balls of solder suspended in flux. You can choose the size of those solder balls based on the size of the apertures in your solder paste stencil. If you're working with very tiny component footprints, you'll need to use a smaller particle size. A rule of thumb is that the narrowest dimension of your stencil opening should be at least five times larger than the particle size of the paste you're using. This will ensure a good release of the paste from the stencil.

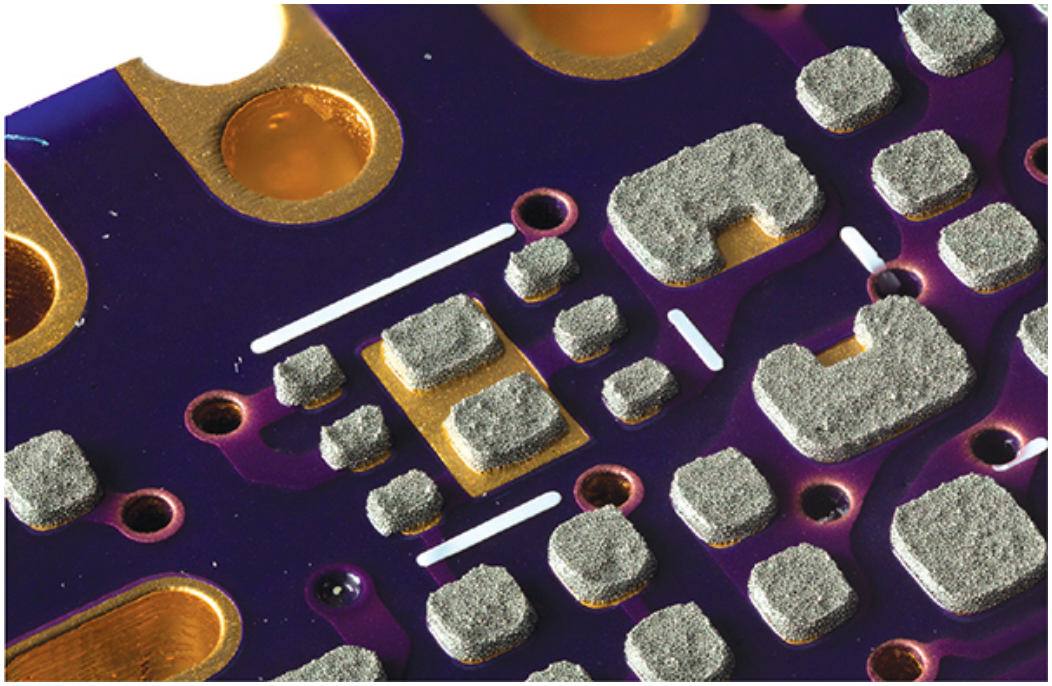


Figure 12-3: Close-up of solder paste, where you can see the solder balls suspended in flux. Image courtesy of Greg Davill.

Make sure you store your solder paste at the right temperature. There's solder paste available that's temperature stable and doesn't need to be refrigerated, but most paste needs to be kept cold and dry. If it's not stored correctly, the flux that holds the tiny balls of solder in suspension will evaporate.

Run tests with solder paste before you do a full production. Make pastes prove themselves before you use them. Your CM or production team will probably have some specific solder paste brands and mixes that they know are reliable and work well with their line.

Wave Soldering

Wave soldering is a process where you put a PCB with components already installed into a machine with a conveyor belt that drags the bottom of the board through a little pool of molten solder. It's a very cost-effective and reliable technique. First, the areas that will be soldered have flux applied. Then the entire board is preheated. A pump creates a small continuously flowing wave of solder that touches all the pads and pins on the bottom of the PCB and solders them as they move through it. The process is most commonly used on boards with through-hole components, but it can be used on boards with SMT parts as well. After wave soldering, you'll need to visually inspect boards for short circuits and manually rework them to fix any you discover.

There are also selective wave solder machines that move the PCB around on a small gantry to position individual component leads that need to be soldered into a small solder wave about the width of a pencil. Miniwave soldering is another related form of selective wave soldering.

If your PCB has certain areas that are thermally sensitive or can't come into contact with solder, you'll need a special fixture, sometimes called a *wave solder pallet*. These are custom-made metal holders that will fit into the machine and shield sensitive areas of the PCB from the molten solder.

To avoid having to wave solder parts, you can use a technique called *paste-in-hole*. This is where you apply solder paste directly on top of through-hole component pads and then insert the component. This allows you to just reflow the entire board, as we'll discuss next.

Reflow

One way to quickly solder lots of surface mount parts is to *reflow* them. There are several reflow methods, but they all have the same initial steps. First, you need to prepare the solder pads by applying solder paste. If you're doing this freehand, use a syringe to squeeze out just enough paste to cover the surface of each pad. You can either use your thumb to depress the plunger in the syringe or take an air-powered approach. Foot pedal–driven regulators designed to dispense glue or solder paste typically cost around $80 and give you easier control over the amount and speed of the solder paste you're putting on. You'll also need an air compressor that can do at least 100 PSI. A bonus of this setup is that it's much easier on your thumb.

A faster way to prepare the solder pads is to use a stencil to apply solder paste to an entire board in one go. Getting a solder stencil fabricated can be a cheap way to improve your DIY assembly process. These are especially useful when you have a board with a large number of SMT components on it. Solder stencils can be made of either steel or Kapton and are designed to be lined up with the solder pads on the PCB. They can be used in a paste printer that automatically applies the solder paste, or you can do it by hand with careful alignment and some practice. You use a

squeegee to spread solder paste across the entire stencil and then remove the stencil so that solder is left only on the board's solder pads. The process is similar to silkscreening.

To improve the reflow of SMT parts with ground pads underneath, it's a good idea to use a technique called *windowing* or *grid printing*. Rather than applying solder paste to the entire ground pad, you apply it in small squares, as shown in Figure 12-4.

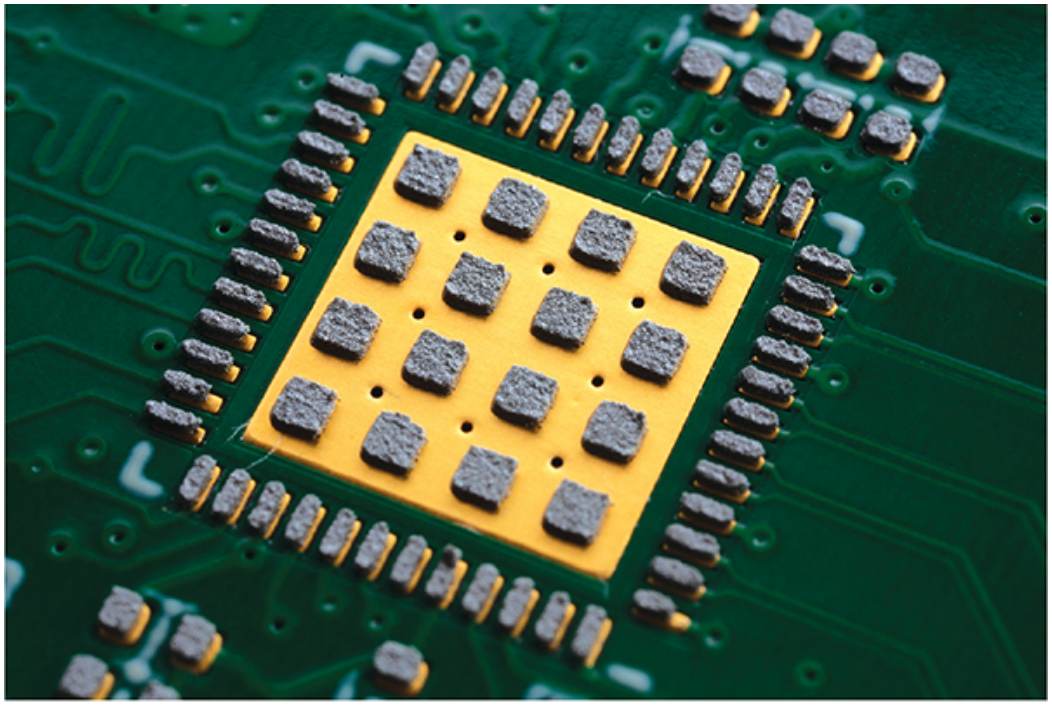


Figure 12-4: Windowing or grid printing on a QFN center pad. Image courtesy of Greg Davill.

Windowing makes it easier for volatiles like flux vapor to escape from underneath the chip. It also helps prevent the chip from lifting off any pads along its edge as a result of too much solder beading together under the chip during reflow.

Solder paste is available in small jars or in syringes. The version that comes in a jar is good for hand stenciling. The version that comes in a syringe is good for manually applying a dot of solder paste to each pad. If you end up using a syringe, use the right size tip. Too large of a tip will make it impossible to control the amount of paste coming out, and you'll end up using way too much. Too small of a tip will take forever to populate a PCB. You can buy packs of various sizes of tips (called Luer-lock tips) on Amazon. It's nice to have 10 or 15 different sizes to choose from.

If you're assembling and reflowing by hand, using expired solder paste is probably okay. It's not going to behave as well as new stuff and will be more clumpy

and harder to work with, but you can usually manage. On the other hand, if you're using a solder paste stencil, new, unexpired solder paste will come off much more nicely and leave cleaner edges on the solder pads. Messy edges can cause too much or too little solder paste to go down, which can cause opens or shorts.

Once you've applied the solder paste, use tweezers to carefully place each component onto the paste-covered pads. Align the parts as best you can, but it doesn't need to be perfect. From here, reflow can proceed in several ways.

One technique is to use a hot-air rework station to heat single components or entire areas of a board. Make sure the air speed isn't so high that it blows the parts away. Hold the hot-air iron directly above the part you want to heat up, and adjust the temperature at the board by lifting the iron higher or lower. You'll see the flux evaporate and the paste start to spread out slightly, and then it will suddenly melt, pull together, and turn shiny. The part will probably move around a little as the surface tension of the molten solder pulls it into perfect alignment. If you have a ground or thermal pad under the part that also needs to melt, you will probably need to wait another couple of seconds for that to begin to flow, too.

You'll probably see some flux start to leak out from under the chip, and then the part will be "floating" on molten solder. To confirm this, very gently tap on any side of the chip with tweezers and see if it springs back into place. Once the part springs back from any perturbation on all sides, remove the heat (by lifting directly up so you don't blow the part out of alignment) and let the part cool. There shouldn't be any solder paste left, and all joints should look fairly shiny.

If any pins look like they missed some solder, apply a little directly to them with a syringe and heat the chip up again. There may also be some larger balls of solder either on the component leads or on the PCB itself if you used too much solder paste. If they're on the PCB, you should be able to just knock them off or pick them off with tweezers, since they will likely be held on only by flux residue. If there are beads or shorts on component leads, either use solder wick to remove them or heat the part again and use an X-Acto knife to push the molten bead off the pin and onto the PCB for easy removal after cooling. The latter requires a fine touch but reduces the risk that you'll end up with too *little* solder, since surface tension will automatically distribute the solder correctly.

Instead of a hot-air reflow iron, you can use hot surfaces or an oven. Hobbyists routinely reflow boards using a skillet or hot plate, for example. Simply place the board with the components sitting on solder paste into the skillet or hot plate, turn it on, and watch the solder reflow. You need to be more careful with temperature here because it's possible to get skillets and hot plates above the glass transition temperature of FR-4, meaning you'll cook the bottom of your PCB. The JEDEC standard limits soldering temperatures to 260°C (for lead-free solder), but your parts will probably survive a little above that. The skillet and hot plate methods

work best when assembling an entire board at once. It's not a great idea to heat the entire board up if you're just trying to work on a single part.

The method that professional assemblers use when building large quantities of boards is a *reflow oven*. There are two main types: infrared (IR) and vapor-phase. Infrared reflow ovens use radiation as the primary method of heat transfer. The PCB runs through several different heating zones. First, the preheat zone slowly ramps up the temperature to prevent thermal shock and keep parts from thermally expanding too quickly. Next, the soak phase increases temperature a little bit more and is designed to get everything up to the same temperature. Then the reflow phase quickly ramps up the temperature to just above the melting point of the solder paste and holds it just long enough for all of the components to actually get soldered. Finally, the cool-down phase eases the board back to room temperature. The times and temperatures that the industry uses for this process are described in JEDEC J-STD-020. Some reflow ovens do this whole process under a nitrogen atmosphere to improve the reflow of lead-free solder. However, IR ovens can have a tendency to heat unevenly and take up a lot of space, since many heating zones are needed.

Vapor-phase reflow addresses these problems and is the latest and greatest reflow method. It uses convection instead of radiation for heat transfer. Perfluoropolyether (PFPE, trade name Galden) is poured into the soldering chamber with the PCBs sitting on a grate above the liquid. The chamber is closed, and the PFPE vaporizes and fills the chamber as heating begins. Heat is transferred through the vapor into the board very evenly as the PFPE condenses on the PCB. Vapor deposition ovens follow the same temperature curve used in IR reflow ovens, but they need only a single chamber, so they take up less space and overcome the uneven heating problems of IR reflow.

While industrial reflow ovens are gigantic and cost tens of thousands of dollars, there are versions available for startups and hobbyists. They're almost all IR reflow, but they work just fine. There's a model from China that some people like called the T962. It's made by dozens of manufacturers, fits on a desk, and usually costs around $200 to $300. You can find it on eBay or AliExpress. There's some replacement, open source firmware for the T962 developed by a hobbyist that supposedly improves its performance, available online at *https://github.com/UnifiedEngineering/T-962-improvements*. They also document some important safety changes, like correcting AC grounding issues and removing some internal masking tape that slowly burns while the oven is running. A company called PCB Arts also offers a small vapor-phase oven called the Vapor Phase One and a slightly larger one called the Vapor Phase Two. They cost about $6,100 and $9,900, respectively.

For a more DIY approach, there are lots of kits available online that let you convert a regular toaster oven into a reflow oven. They all basically do the same thing: Run a PID loop to hit a particular temperature curve by looking at a temperature probe inside the oven and switching the oven heater relay on and off. An even simpler approach is to just use an unmodified toaster oven. I've successfully reflowed many boards in a completely stock toaster oven that I bought simply because it was the cheapest one in the store. It's rare for toaster ovens to even be able to get above 260°C, and hitting an exact temperature curve isn't necessary for one-off prototypes. However, you may see better results with more feature-complete reflow ovens, especially as you reflow more and more boards. And it hopefully goes without saying that you should never put food in a toaster oven again after it's been converted into a piece of electronics assembly equipment.

Assembly is much easier if you've designed your boards to have parts only on one side, since the board has to go through the reflow process only once. If there are parts on both sides, you'll need glue or epoxy on one side to keep the heavier parts from falling off once the solder melts again when the parts are upside down (small parts like resistors and capacitors will stay in place without glue due to surface tension). If you need parts on both sides, you can use something like Loctite 348 to keep the parts attached to the board. In a production environment, this glue can be dispensed in a similar way to solder paste and at the same time. The heat from the reflow oven will cure the glue. When the board goes through again, but upside down, the solder will melt, but the glue won't. The solder won't drip off because surface tension will hold it in place.

Safety

To stay safe and comfortable while soldering, use a fume extractor. Solder fumes don't contain lead, but some people are sensitive to flux vapors. You can buy very cheap fume extractors that are basically just a small fan blowing into a filter (usually of questionable efficacy), or expensive extractors that have a flexible hose running back to a HEPA filter. The expensive version will actually remove the vapors from your work area, but dispersing them is often enough to keep you from breathing them in directly and keep you more comfortable.

The biggest risk of lead contamination in soldering comes from touching and ingesting solder. When wiping down your bench, use a wet cloth instead of a dry cloth. A wet cloth will capture the solder particles, whereas a dry cloth can kick them up into the air or spread them. Don't eat or drink when you're soldering, and always make sure to wash your hands when you leave the bench to go do something else.

Cosmetics can contaminate your PCB and make soldering more difficult. If you're reworking a customer device or something that's high reliability, don't wear

cosmetics or lotions. If you touch your face and then touch the PCB, those chemicals can transfer.

Tips for Assembly

If you aren't doing the assembly yourself, you'll likely have a trained assembler doing it for you. Maybe a pick-and-place machine is doing most of the work, but there's a good chance a person is still somehow involved along the way. Treat this person with the utmost respect, and listen to their suggestions. They likely have way more experience than you in assembling electronics, and they're intimately familiar with how the process works. If they suggest changing the assembly procedure, consider their advice carefully. There's a very good chance they'll be able to help improve assembly speed and accuracy. The other reason you should be very nice to assemblers is that you'll find yourself relying on them to get you out of jams. If you screwed up your design and need it to be reworked, they'll do it. If you need your job rushed, they'll be the ones working overtime.

Even if you're hiring an assembler, you should still assemble the first unit (sometimes called the *engineering model*) yourself and use it to figure out the order of assembly. Going through the process yourself will help ensure there won't be any impossible steps, like screws that can't be tightened because of insufficient tool clearance. If there are subassemblies, keep in mind that they need to be tested before being integrated into the final assembly. Look out for possible mistakes, like parts that could be rotated incorrectly or forced to fit in the wrong place.

If you're assembling products en masse, consider doing a *time study* to figure out where your assembly inefficiencies are. Observe your assemblers as they work and identify the parts that are slowing them down, parts that can be automated, and other preparations that can be done prior to assembly that will speed things up. Make sure all of their equipment is sufficient for the job, is being used correctly, and is functioning properly.

Well-trained assembly and repair technicians should know when they need new equipment and how to correctly perform rework. Having assemblers trained by an organization like IPC is a good way to make sure they know what they're doing, but lots of good on-the-job experience is what's most important. If you're a startup or individual without the money for expensive training or certifications, you can sometimes find IPC documents online or on eBay. You won't be certified, but those documents will show you how to perform reliable rework. As you can probably imagine, assembly is very hands-on, even for people who aren't doing the actual assembly. Reliable assembly processes require attention to detail and clear, continuous communication with the assemblers.

Documentation

Once you've ironed out the assembly process by creating the engineering model yourself, prepare a step-by-step instruction manual that describes how to assemble the device. An easy way to format an assembly document is as a PowerPoint presentation. That way, assemblers can view each step full-screen on their monitor and press a single button to move to the next step without scrolling around.

The instructions document should be clear; have plenty of diagrams, drawings, and pictures; and include quality checks so the assembler can be sure each step was completed correctly. Each step should have a list of the tools and equipment needed to complete it. Use notes or callouts to specify tolerances, torques, maximum and minimum acceptable values, and whether any equipment needs to be calibrated first (like a network analyzer), as well as obvious information like the part numbers to use. Specify the color, gauge, and core type (solid or stranded) of any wires that are soldered in as part of the assembly process. If you use any cable ties or cable lacing, specify where and what size. Wire termination also needs to be specified. Does it need heat shrink? Does it need to be crimped, soldered, or something else? When the instructions are ready, you can pass them off to your assemblers on your factory floor, but be sure to update the document if certain steps need to be clearer or if there are any problems or changes to the product.

Of course, the assembler also needs the BOM for your design. This should contain *all* the parts necessary for assembly, not just the parts that get soldered to the board. For example, if you need thermal compound, heat sinks, or jumpers, those need to be included.

Another useful document is a *travel sheet*, sometimes called a *traveler*. This is a document that follows the PCB during the production or repair process to help track inventory and manage quality. It's a checklist of everything that needs to be done to the board, when it happened, and who did it. You can implement a travel sheet digitally with a spreadsheet, serial number, and barcode scanner, but having a physical piece of paper that people have to sign is much harder to forget about and easier to keep up to date. Travel sheets help ensure that procedures are being followed correctly. If a device fails unexpectedly in the field, you can use its travel sheet to determine whether any steps were missed or performed incorrectly.

Tools

Using the right tools makes all the difference during assembly. For example, when you're assembling anything with RF connectors (even when testing and experimenting in the lab), use a torque wrench designed for the connectors at hand. Different connectors need different torques. Using the wrong torque wrench can overtighten and break the connector, and simply finger-tightening can result in insufficient torque and higher signal loss. Unless you're working with very high-

frequency signals, you'll likely just need an SMA torque wrench, since that's the most common screw-type RF connector. You may also want an N-type wrench, since that's what most test equipment uses.

Inevitably, something during assembly or prototyping will require tape. If you're new to electronics, you may assume that electrical tape is the obvious choice here. You'd be wrong. Electrical tape is the worst. It leaves a sticky black residue, doesn't stick well to begin with, and is messy, imprecise, and doesn't work well at high temperatures. Instead, use polyimide tape. Sometimes known by the 3M trade name Kapton, this tape is a wondrous thing. It doesn't leave residue, is electrically insulating, and can survive high temperatures (even direct application of a soldering iron). Every electronics lab should have at least one roll around. Its most common uses are for taping wires in place and for providing a layer of insulation between conductive surfaces.

Similar to Kapton tape are so-called *circuit dots* or *circuit tape*. These look like small stickers that come on a sheet and can be peeled off and used to fix wires into place. Their most common application is when a PCB has been reworked and has a "blue wire fix." To keep the repair clean and to keep the wire from being hung up on another part of the assembly, circuit tape can stake the wire down at certain points on the PCB. Circuit tape doesn't leave residue but has a stickier adhesive than Kapton. It's not as tolerant of high temperatures as Kapton but can be easier to work with in tiny pieces. This adhesive also has a cure time of about 72 hours, meaning it will increase in strength after it's applied. Circuit tape generally comes in two forms: small circles and a butterfly shape. The butterfly shape is specifically designed to hold down wires along its length. Circuitmedic.com sells these products.

Another kind of tape that's helpful during assembly is copper tape. It can be used to tune antennas, as shown in Figure 12-5, or as a shield.

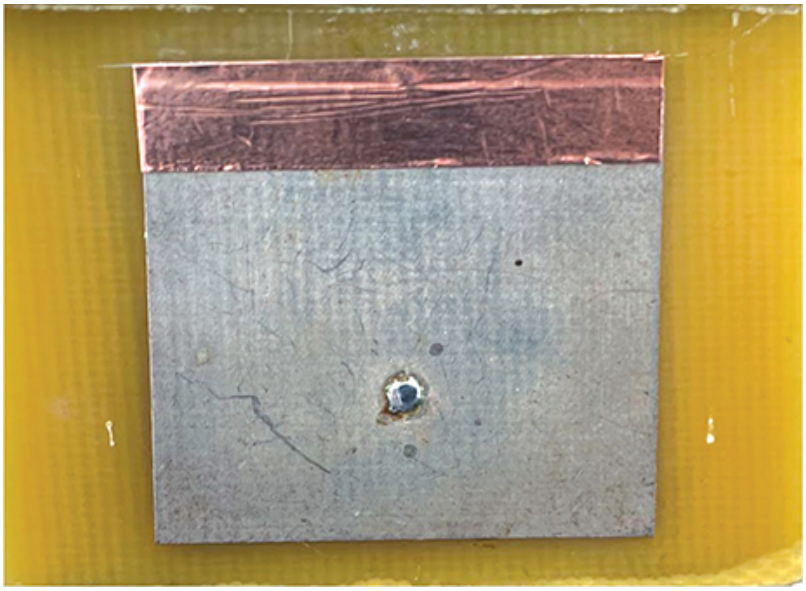


Figure 12-5: Copper tape was used here to change the response of this antenna. Simple trial and error with a network analyzer dictated where the tape should go and how much to use.

Copper tape is easy to solder and typically has a conductive adhesive, but be aware that the adhesive isn't nearly as conductive as the copper itself. It's useful to have around because it can let you easily assess shielding effectiveness during EMI testing and can even be used to replace ripped traces or pads.

Conformal Coatings and Potting Materials

If your device will be subjected to particularly rough conditions during use, consider using a conformal coating or potting on the PCB. *Conformal coating* is a thin, clear liquid polymer film that completely covers a PCB. This prevents dust, moisture, and other environmental contaminants from affecting the PCB and components. It can also help prevent arcing between high-voltage components that are close together and can prevent tin whiskers from forming. The advantage of conformal coating is that it's still possible to repair and rework the board and see all silkscreen and components after the coating is applied.

You need to make sure the coating you choose can survive the entire temperature range of your specifications. Table 12-3 summarizes the temperature ranges and other qualities of different conformal coating materials.

Table 12-3: Comparing Conformal Coatings

Conformal coating	Temperature range	Condensation rating	Humidity rating	Tin whisker mitigation	Difficulty of rework

Conformal coating	Temperature range	Condensation rating	Humidity rating	Tin whisker mitigation	Difficulty of rework
Acrylic	–65°C to 125°C	Good	Excellent	Good	Low
Epoxy	–40°C to 125°C	Poor	Poor	Excellent	High
Parylene	–65°C to 200°C	Excellent	Excellent	Excellent	Very high
Silicone	–70°C to 200°C	Excellent	Poor	Good	High
Urethane	–65°C to 125°C	Poor	Good	Excellent	High

Acrylic conformal coating is easy to remove using a mild solvent like acetone. Urethane and epoxy coatings are typically very hard, meaning they're more prone to cracking than other coatings. They're also quite difficult to remove and can outgas, making them unsuitable for aerospace applications. Silicone can be removed with abrasion. It can also outgas, especially RTV (room-temperature vulcanized rubber). If your silicone isn't fully cured, the materials that get outgassed can corrode your PCB. Parylene is great at protecting your board, but it has to be deposited in a vapor deposition chamber under vacuum. Once applied, it's almost impossible to remove. If you *do* manage to remove it, you'll need to recoat the area with a new conformal coating material because parylene doesn't stick to itself very well.

There are a few special cases where you can't use a conformal coating. For example, a barometer IC has a small hole in the top to sense ambient pressure. If you apply a conformal coating over this IC, it won't work. You need to take care to ensure that all parts are able to be conformally coated. In general, if a part isn't marked as "no-clean," it can be safely coated. To maintain maximum repairability, it's not recommended to conformally coat a PCB unless you really need to. Try using a sealed enclosure instead. Your repair technicians will thank you.

An alternative to conformal coating is *potting*. Potting material can be either a rubbery or rigid resin that fills the enclosure holding the PCB or is molded around the PCB. Potting material can be useful in deterring reverse-engineering efforts: It's difficult to remove, making it hard to view components and traces and access signals. However, this also means that it's very difficult to repair or rework a PCB

after it's potted. Some potting material can be removed with a solvent like acetone or by heating and plucking the material off with pliers.

You need to make sure there aren't any air bubbles in the potting material if the device will be exposed to a wide range of pressures. Air pockets can expand and pop, damaging the board. Outgassing is another possibility you'll need to evaluate in the case of products used in aerospace applications.

Some potting materials, adhesives, epoxies, and soft gaskets have a curing time associated with them. This can have a big impact on your assembly time if you have to wait for the material to fully cure before moving to the next step. If you get impatient, your material might not get cured all the way and can fail prematurely. One way to alleviate this problem is to use *light-curable materials (LCMs)*. These materials cure very quickly by shining UV light on them and can immediately be inspected. Using LCMs also removes the need to correctly mix multipart epoxies. Of course, you need to be able to get light to whatever you're trying to cure. That may mean making the channel that will contain the LCM transparent.

Experiment with different LCMs to find a material that works well with the materials you're bonding together and that cures in the right way (soft or hard). Carefully follow the instructions and recommendations of the manufacturer; they'll suggest certain curing lamps and dispensers to use. You also need to make sure your curing lamp meets the minimum brightness requirement to completely cure the LCM. You can do this by using a radiometer to measure your lamp. This would be an important step in an assembly process. Keep in mind that if your dispenser isn't opaque to light, your LCM will cure inside the dispenser and clog it up.

Cleaning and Storage

After assembly, PCBs are normally cleaned with a solvent like isopropyl alcohol. Some components, however, have sensitive materials or structures that won't survive a cleaning process. Such components will be marked as "no-clean" in the datasheet. If you're hand-cleaning boards, you can just avoid any sensitive components. However, if you're using batch or automatic cleaning, you'll need to skip the cleaning step to prevent any damage.

Parts that aren't stored properly can fail in strange ways. For example, too much moisture in an IC package can result in a phenomenon called *popcorning*, where the water trapped inside the chip expands and causes voids. To prevent this, store components in humidity- and ESD-controlled environments before use. You can use something called a *dry box*, which is exactly what it sounds like: a purpose-built box that protects components from moisture. All components have a floor life, which is the amount of time they're rated to spend on the factory floor before they'll have absorbed too much moisture and will cause assembly problems. Once

the parts come out of their sealed bags, the timer starts. By placing components in a dry box, you can pause that timer.

If you exceed a part's floor time or if parts are exposed to higher humidity, they need to be *baked out* before assembly. This involves putting the parts in an oven to slowly evaporate out any moisture that's migrated into the IC package. To prevent melting the plastic tape and reels or other packaging, baking is typically done at low temperatures for a long time—anywhere from 3 hours to 79 days, depending on the component and how it's packaged. IPC/JEDEC standard J-STD-033D contains tables you can use to figure out the temperature and time required to safely bake out your components.

All of that said, for small-scale manufacturing or prototyping, you don't need to worry about getting a dry box or baking out parts. These tools and techniques are really only important when you're working with a large number of parts and need high assembly yield. The easiest way to keep parts in a humidity- and ESD-controlled environment is not to open the packages they come in until just before assembly. Cut only a small slit in the top of the packages so you can store unused parts back in the same bag with the original desiccant packet and humidity card. Alternatively, you can buy ziplock ESD bags and reseal parts in them. You should also keep PCBs stored this way. Ziplock ESD bags are cheap and can be reused with different batches of parts.

To help eliminate ESD problems during assembly, you can use an *alpha ionizer*. This device attaches to your compressed air source and ionizes the air coming out using a small amount of radioactive polonium. The radioactive material itself doesn't come out, and it's completely safe to use. Alpha ionizers are handy because you can simply spray your board with ionized air to remove any debris or static buildup. Using an air compressor *without* an alpha ionizer to blow debris off your boards is actually bad, since compressed air is especially vulnerable to ESD problems. As the air rubs against insulating surfaces inside the compressor and against other air molecules when it exits the nozzle, it can build up a charge, which can collect on your PCB and cause ESD damage. Alpha ionizers prevent this by creating a roughly equal number of positively and negatively charged particles in the air that leaves the nozzle, which keeps charge from piling up and creating an ESD problem.

Inspection and Testing

Inspection is a critical part of assembly. If you're using an external assembler or turnkey service, it's their responsibility to do the PCB inspection before they begin assembly. If you're assembling boards yourself, carefully inspect the fabricated boards before you start soldering them together. There's nothing worse than

spending a lot of time soldering a board only to discover there was a fabrication error like the one in Figure 12-6, where the PCB should have been rejected.

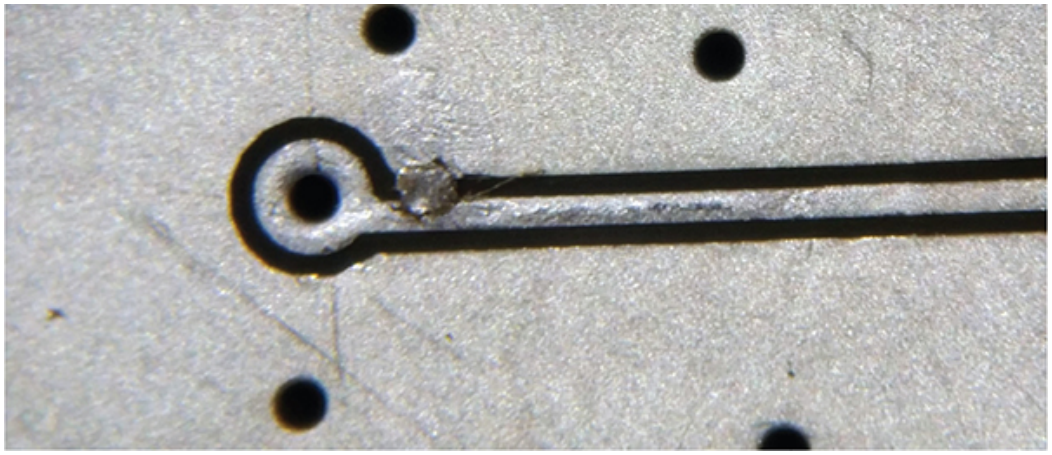


Figure 12-6: A thorough visual inspection under magnification will catch fabrication errors like this short circuit.

Whenever I get a board in (assembled or not), I always take a picture of it before I do anything else. For bare boards, a flatbed scanner is a great way to capture the board in high resolution. That way, I can always go back and see if an issue was present when the boards arrived, or if it was incurred during bring-up and testing. If you see a problem, tell your fabricator! They can help diagnose the issue. Reputable manufacturers will offer to refabricate the design for free if they've made any mistakes. Having pictures of the boards before anything was done to them is also a great way to compare manufacturers. You can check their tolerances, finishes, and quality to inform future decisions on which fabricator to use.

Testing is also an extremely important part of the product manufacturing process. In fact, Chapter 13 is entirely dedicated to the topic. For now, be aware that your PCB fabricator should do a 100 percent electrical test before shipping your boards. That means they'll verify that the electrical connections on the PCB are exactly the same as those in the CAD. After you assemble the board, you'll also need to perform functional tests to make sure all the components are working and all solder joints are good.

If your design depends on high-tolerance ceramic capacitors, you'll need to wait at least 48 hours after assembly before testing. For example, if you have a high-performance filter or sensitive analog front end that relies on ceramic capacitors being very close to their rated value, you'll see the expected tolerance only about two days after reflow. This is because ceramic capacitors undergo an aging process that can affect their capacitance. Barium titanate molecules that make up the dielectric material of the capacitors will rearrange themselves during heating, and this process will continue for years after the initial heating. However, the largest

change in capacitance resulting from this effect occurs within the first 24 to 48 hours. If you're using ceramic capacitors just to bypass your IC power pins, this waiting period isn't necessary.

System Integration

System integration is harder than you think. After all of your individual boards are assembled, tested, and working, you may be under the impression that system integration is as simple as connecting them together. However, even if you've managed to test all of your boards together, the task of stuffing those guts into an enclosure and having them all still work can be surprisingly difficult, so make sure you leave plenty of time in your schedule for this process. The trick to a smooth system integration is planning and incremental testing.

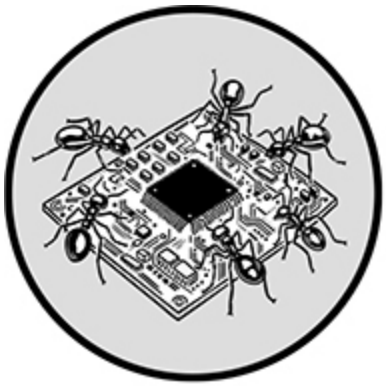
Planning starts all the way back in your requirements and specification documents, when you design the interfaces and connections between different parts of the final product. Don't forget to think about wire harnesses and cable management. You'll need to work closely with your mechanical engineer(s) to design how all of the parts come together, where they go in the enclosure, and how they're installed.

Once you have all of your parts ready for integration, make sure they're tested individually first. Then gradually add subassemblies one by one, and finally test the entire system together. Working piecemeal will allow you to catch problems early and avoid having to take apart your entire system to figure out what's going wrong. It can also prevent you from accidentally damaging another part of your system if one subassembly fails in a catastrophic way. It's a good idea to test your subsystems on bench power supplies at first, so you can make sure your current consumption is what you expect and that everything is working properly.

Conclusion

In this chapter, we talked about how to get your parts soldered onto your PCBs, whether they be a few prototypes that you solder together yourself or a large production run assembled in a factory. Understanding how assembly works and where the pitfalls are will help you design electronics that are manufacturable and reliable. Getting good at soldering is just like getting good at anything else: You need to practice. Having that skill will also make you great at rework, which can save you significant time and money by allowing you to hack up or fix the prototype you have rather than pay and wait for a second version to be fabricated and shipped. Personally, I find soldering very relaxing, so I encourage you to grab your soldering iron and assemble something.

13
TESTING



Testing is how you find out if the electronics you've designed actually work. It's an important moat that separates an engineering prototype from a device good enough for the public to use. In this chapter, we'll talk about the kinds of testing you may need and how to choose and design those tests. It may be tempting to skimp on testing so you can get your product out the door faster, but this will always backfire and permanently affect what customers think of you. Effective testing ensures that your product has a good reputation and that it can pass regulatory requirements.

The Philosophy of Testing

NASA's Jet Propulsion Lab has a saying: Test like you fly, and fly like you test. This means you should design your tests to mimic the actual environment where the device will be used, and you should use your devices only in a way that's consistent with how you tested them. If your device will be used in cold temperatures, test to those actual temperatures (plus a safety margin). If your device was never designed or tested for use in cold temperatures, then don't expect it to work under those conditions.

The catch to this philosophy is that you need to correctly anticipate your device's operating environment. If you start to see a lot of units fail under conditions outside of what you tested, you may have made an incorrect assumption about your users, the environment that your product is used in, and how people are using it. These are all product questions, and changing product requirements is a big deal. But if your units are passing internal tests and failing in the field, consider that the problem may be that you aren't testing your product the way it's actually being used.

For any product to succeed, you need to prove that you can actually achieve all of your product requirements. To do this, testing is absolutely critical. But testing in and of itself isn't enough; the testing must be complete and relevant. The first question that testing answers is "Can it?" The second question it answers is "For how long?" You need both parts of the puzzle for the user to understand the device's reliability. The answers to these questions also matter to you, the manufacturer, because they'll tell you how often to expect repairs to be necessary. They also tell you what the product's expected lifetime is, which is important for customer support and end-of-life purposes.

All components have an expected lifetime. This is typically described by something called the *bathtub curve*, in which the failure rate is highest at the very beginning and very end of a product's lifetime. Figure 13-1 shows a typical bathtub curve.

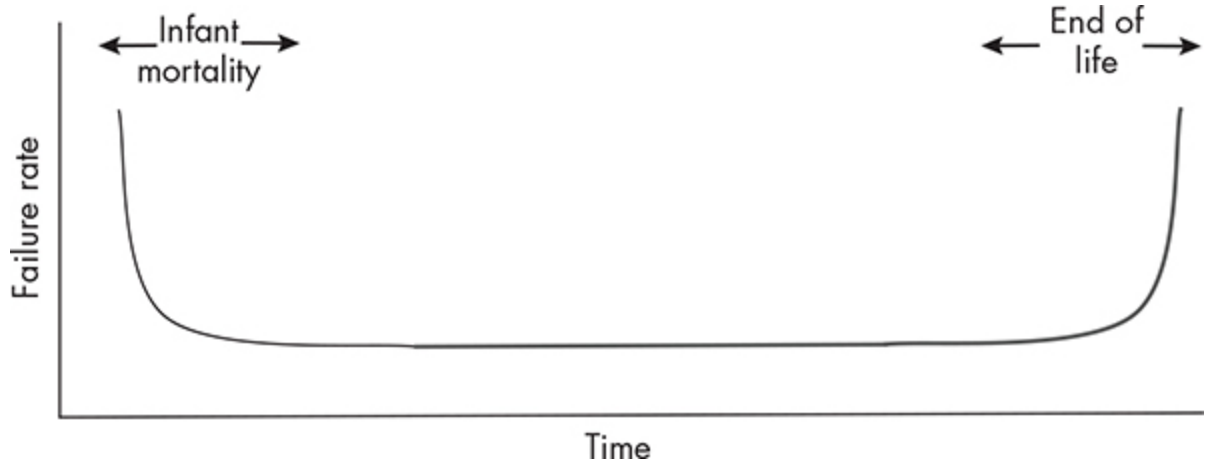


Figure 13-1: The bathtub curve plotting a part's failure rate over time

At the beginning of the bathtub curve, parts with manufacturing problems will fail early during acceptance testing. The goal is to design your tests so that all components with manufacturing defects fail before the product goes out the door. In the next part of the bathtub curve, parts continue to function for their normal lifetime. Finally, parts start to fail at the end of their expected life.

Figuring out what this bathtub curve looks like for your product is important because it will drive your test plan and your warranty plan. If you don't test enough, you may miss some of the early failures, which will lead to devices failing early in the field. However, if you test for too long, you'll be wasting time. If your warranty covers failures well into the expected end of life of the parts, it's going to be really hard to make money since you'll be replacing products that have reached the natural end of their lives instead of only products that failed unexpectedly early.

Regulatory Testing

In addition to the tests you design to validate your own product, there are tests that governments or industries use to independently ensure that you designed your product to be safe and reliable. Throughout the 20th century, as electronic devices became more ubiquitous, they also started interfering with each other. This wasn't just a matter of radios interfering with other radios: A security guard using a walkie talkie at a hospital could accidentally cause medical equipment to malfunction (see Stephen Lapinsky and Anthony Easty's article "Electromagnetic Interference in Critical Care," published in the *Journal of Critical Care* in October 2006, for details). To make sure all equipment got along, whether it was intending to transmit a signal or not, governments around the world started requiring electronic products to be tested and prove that they wouldn't cause interference in other devices and that they would still function even if other devices created interference.

Meanwhile, standards organizations also started developing methods to prove that a product was safe, leading to lengthy and detailed standards documents that products can be designed to meet. If you need the full text of such a standard, you'll quickly find that buying it from

the standards organization is extremely expensive. Luckily, there are several tricks for getting them cheaper. If you need a European standard, you can buy it for less from the Estonian Center for Standardization and Accreditation (*https://www.evs.ee/shop/*). You can also Google the name of the standard and add *filetype:pdf* to limit the results to only PDFs. This will often return drafts or slight variations of the standard you're interested in. You can usually download draft versions of standards for free once you sign up on the standards organization's website; however, they often remove drafts after a while, so you can't always find what you need this way. Finally, you can search eBay for hard copies of standards, which are often significantly cheaper than buying a digital version from the standards organization.

Approval Marks

If you look closely at almost all electronic devices (usually on the bottom), you'll find several logos that indicate which tests the device has passed. As an example, a *CE* mark indicates that the device is "Conformité Européenne," meaning it conforms to European safety and environmental standards. This mark is required if you want to sell your device in Europe. To get the CE mark, you need to identify the relevant directives, identify the harmonized standards, figure out which requirements apply to your product, decide whether you need an independent conformity assessment, and actually run the tests, all the while keeping correct documentation of everything.

While it's possible to do all of that yourself, you shouldn't. Instead, talk to a testing lab. They're experts and will tell you what applies and what doesn't. There are lots of directives to look through, with some of the most commonly required being RoHS (which limits hazardous substances you can put in a product) and WEEE (an electronics recycling directive). If you want to get a quick idea of what directive may apply to you, look up products similar to what you're building and see what directives applied to them. Searching *DOC* or *declaration of conformity* will usually return a short summary of the requirements and tests a product had to pass.

Another mark you'll commonly see is the *UL* mark from Underwriters Laboratories. Many people are surprised to learn that UL is a private company and that there's no legal requirement to get their sign-off. Nevertheless, people pay for UL approval because the company is one of the oldest and best known testing labs in the US. They'll perform consulting services to help you determine what regulatory testing is required, and they'll actually perform those tests. This includes safety testing, which is what they're most known for. Since many retailers won't carry a product unless it's been safety tested, getting UL approval is often a practical requirement, if not a legal one. If you plan to seek UL approval, make sure you talk to them before you lay out your PCB. They'll give you certain trace and space requirements and have minimum dielectric thicknesses. It's important that you know those things going into layout so you don't end up submitting a completed, working PCB to UL only to have them tell you that you need to redesign it.

Electromagnetic Compatibility

Not only must electronics be safe for people to use, but they must also be safe to use around other electronics. *Electromagnetic compatibility* is a general term that usually refers to both electromagnetic emissions and electromagnetic immunity. *Electromagnetic emission* is intentional or unintentional radiation from a device. Devices must not produce unintended radiation above a certain level because it may cause other devices to malfunction. *Immunity* is the ability of a device to function in the presence of such radiation. Alternatively, the term *susceptibility* describes a device's inability to function in the presence of interfering fields. The complete details of international electromagnetic emissions laws are outside the scope of this book, but make sure you know which laws apply to your product and how to comply with them.

All electronics devices that are sold as products must undergo regulatory emissions and immunity testing. In the United States, the FCC specifies these tests and approves devices. Navigating the set of laws governing all of this can be complicated, but it's also not your job! There are plenty of testing labs that can help you figure out which tests

you need to pass. It's a good idea to reach out to several testing labs to get a quote. This ensures that you'll get a fair price, but it's also a free way to get multiple experts to judge what your testing requirements actually are. That way, if one of them makes a mistake and removes or adds a test that another lab said you did or didn't need, you'll know to ask about it to help make sure your product is being tested correctly.

You can schedule a meeting with UL to talk to an engineer who will answer any questions you have and tell you exactly which regulations are relevant to your product (not just electromagnetic compatibility rules, but other safety and environmental regulations, too). It's best to start asking questions as early as possible. You might think you need to wait until a product is ready to test to engage these people, but that's almost always too late. The situation you really want to avoid is having a product that's 90 percent done, only to discover that your EMC specifications were wrong and that you have to do an expensive and time-consuming redesign late in the design cycle. Some EMC labs report that around 50 percent of tested products fail EMC requirements on the first try. You can avoid this if you engage with experts early. Talking to experts early is also valuable if you're trying to distribute your product in multiple countries. An EMC or UL consultant can help you navigate your unique set of multinational regulation specifications.

Similarly, instead of waiting until your product is in production to run any of the actual EMC tests, you should take advantage of the prescan tests that most labs offer, which can be done on engineering models or other prototypes. These will save you money because the tests themselves are less expensive than the final regulatory tests and because it's much cheaper to fix EMC problems caught early in the design than those caught during production. Prescan tests collect all the same data that the FCC requires for approval; it's just that the results aren't submitted to the government. An example rate for a test like this is about $200 an hour, which includes full use of an antenna chamber and a technician's time to help you set up and run the tests. Once the tests represent your product in its shippable form, you can actually submit the data from your prescan to the FCC, along with a lot of paperwork, and use that as your approval data. To give you an example

of the cost of getting full FCC approval, a company like UL will charge around $15,000 to $20,000. You can end up paying more than this depending on the number of countries you need certification for and how complex your product is.

It's important to run EMC tests under realistic conditions that match how the product will actually be used. Use your real enclosure and your real cables. Run all of your electronics in their worst-case mode. For example, if you have a transmitter that has an adjustable power output, run it at the maximum setting. Turn everything on so the device is using the maximum current possible. The easiest way to do this is to write a separate version of your firmware for EMC testing.

When running immunity tests, you need a way to determine if your device survived or not. The same firmware you use for EMC testing can also include code to exercise the functionality of each part of your device. For example, you can display a test pattern on screens, blink LEDs, read and write to memory, check sensor outputs, and so on.

Sometimes it can be useful to look up the FCC submission of another product for comparison. The easiest way to do this is to use the website *http://fccid.io*. Simply type in the name of the product, the company, or the FCC ID (found somewhere on the outside or perhaps in the battery compartment of the device), and the entire submission will come up. Companies are allowed to ask the FCC to withhold data about their testing until the product is actually released, so you may not be able to see everything immediately.

It may seem like EMC is just a regulatory hoop to jump through, but it can matter for prototypes, too. Your device can actually interfere with itself. For example, a poorly routed PCB that has high unintentional emissions can interfere with a Wi-Fi, GPS, or Bluetooth radio on the same board and prevent things from working.

Battery Certification

All lithium batteries or products that contain lithium batteries must be UN38.3 certified. The United Nations considers lithium batteries to be "dangerous goods," and various international agreements require that

you pass the tests in UN38.3. These include thermal, vibration, shock, external short circuit, impact, overcharge, and forced discharge tests, as well as altitude simulation. A testing company can do all of this for you and get you certified so you can ship your product.

Medical Device Approval

One of the biggest regulatory burdens comes when trying to build a medical device. The amount of red tape you need to navigate to get a medical device on the market is daunting, as it should be. Medical device regulatory approval is outside the scope of this book. To determine what applies to you and what your design requirements need to be, talk to a field-specific consultant. For example, UL offers a service where you sit down with several regulatory experts, describe your device, and determine exactly what you need to do to get it approved and on the shelf. The cost of this service is usually something like $600 to $1,000, depending on how long the meeting takes. Of course, UL will also perform the required tests for you and submit the right data and documentation on your behalf.

PCB Functionality Tests

The first line of defense in preventing defective products from getting shipped to customers are the tests the factory does to determine your PCBs were manufactured correctly. Before assembly, they'll often perform a *100 percent electrical test* using something like a flying probe. This is a machine that quickly moves probes to each of the pads on your board and runs continuity checks to make sure there are no accidental short or open circuits. After assembly, manufacturers usually put your board into an automated optical inspection machine for visual inspection. This machine will automatically recognize defects like tombstones, misaligned parts, or missing components. If your design has BGA components, it will likewise go through an automated X-ray machine to detect solder shorts and other issues that aren't otherwise visible.

Board Bring-Up

The process of testing a newly assembled PCB for the first time is often called *bring-up*. During this process, you'll make sure there aren't any major mistakes on the board and exercise all of its functions. Start by visually inspecting the board, looking for parts that have been rotated, missing parts, missing solder, shorts, and the like. Next, use a multimeter to check continuity between your power planes and ground. A dead short there can damage components. Note that some RF components may make it look like there's a DC short between power and ground during a continuity check, even if there's not. If you have a development board of the suspected RF component, you can perform the same continuity check and verify that it's expected.

If you designed your board as separable subcircuits, bring up one sub-circuit at a time. This will make it much easier to identify where any potential problems lie, and it can prevent a single misbehaving circuit from killing your entire board. To facilitate this process, remove any jumpers, 0 Ω resistors, or connectors that connect subcircuits together, and add them back in one at a time as you verify each subcircuit.

Next, set up a current limit on your power supply. Give yourself a little room above the calculated current consumption and power it on. That way, if anything goes wrong, you'll minimize the damage since the power supply will go into constant current mode and start dropping voltage. If you see more current drawn than you're expecting, don't run the board for too long. If that extra current is being consumed because of a short or other problem and is turning into heat, you might damage a component.

A thermal camera is a great way to figure out if any particular part is consuming more current than it should be. You usually need to power the board on only for a second or two to see which chip is getting way hotter than any others. Traditionally, thermal cameras have been really expensive, but you can now buy one that plugs into a smartphone for a few hundred dollars. If you can't afford a thermal camera, or can't get one in time, put a layer of clear plastic wrap over your PCB and power it on. The hot chip will slightly melt the plastic wrap, and you'll be able to instantly tell where the problem is.

After you've verified that the current consumption looks correct, nothing smells burned, and nothing is getting too hot, use a multimeter to check the voltage on each power plane and voltage regulator. The next step is to try programming any programmable ICs. You should also do a fit check of any enclosures or mechanical components. From here on, it's mostly a software exercise to test each component. Try reading and writing to any memory; talking over SPI, I2C, or any other buses; and capturing inputs and writing outputs.

These methods work for bringing up your first boards, but they aren't scalable when you have hundreds or thousands of boards to test. Once you've fixed all the bugs in your design and can reliably bring boards up by hand, it's time to automate the process and run your test plan.

Test Fixtures

The infrastructure allowing you to run your test plan is often fairly extensive. This includes the software and hardware needed to exercise 100 percent of the functionality of your product. To paraphrase Andrew "bunnie" Huang, the test infrastructure is the product that lets you test your product. Don't underestimate how much time and engineering it will take to design and build a high-quality test jig and the accompanying firmware. A test jig usually performs tasks like programming ICs, running in-circuit tests, boundary scanning (like JTAG), verifying memory (by writing and then reading each bit to check that the write worked), measuring voltages and currents, and even pressing buttons or checking LEDs.

The most common way of testing a PCB in production is with a jig called a *bed-of-nails* jig. It looks kind of like a panini press, and the PCB sits inside it while the lid is shut. Special pins called *pogo pins* make contact with test points. These pins come in several varieties, but they all have spring-loaded tips that can press into the test points when the bed-of-nails jig is closed. You can find pogo pins with a spear head, crown head, needle head, or cupped head. Spear and needle heads are good for contacting flat test pads, crown or star heads are good for contacting untented vias or unpopulated through holes, and cupped

heads are good for contacting the remaining lead of a soldered through-hole component. Pogo pins will slowly wear out over time, so make sure to use receptacles to hold your pogo pins so they're easy to replace. If you solder wires to all of them rather than using press-fit receptacles, your test jig will be a lot more painstaking to maintain.

Other common parts you may find useful for building fixtures are fixture clips, pogo pin probes, and toggle clamps. The company Merifix sells a lot of parts and kits that are useful for building a bed-of-nails fixture. You can also find parts from a company called INGUN and another called QA Tech.

A typical bed-of-nails fixture can cost $5,000 to $10,000 or more depending on the features. However, once you pay for the fixture, it's much cheaper to reconfigure it for different PCB revisions by just making a new plate for the pogo pins (usually a couple hundred dollars). You can also try to save money by buying used bed-of-nails fixtures from eBay or electronics surplus stores and reusing parts of them. If you pay someone to build a fixture for you, they'll need coordinate files that locate each test point, wiring diagrams, pinouts for the connectors on the back plate, drawings for special features like switch actuators, and cutouts for tall components.

Just as important as the tests themselves is the human side of testing. As you're designing your test jigs and writing your test plan (which is covered in the next section), think about how the people executing the tests will run them. Your test jigs need to be reliable, updatable, intuitive, and require as little human interaction as possible. The details of how to come up with a good user experience are outside the scope of this book, but as an example, consider symbols instead of text in your interface. Different people from different parts of the world may be using your jig, depending on where you're manufacturing. An ideal test jig requires only two human interactions: putting the board into the jig and pressing Start, and removing the board when the test is done. Automation should be as complete as possible.

Once you have a great testing plan and testing jig, you need to track which boards have been tested and their status, and you need to verify that all of the tests are actually being run correctly. Less scrupulous

factories may sometimes try to cheat by passing devices that weren't tested or that failed in order to artificially boost production yield. This is relatively rare, but mistakes or poor recordkeeping can also result in devices getting shipped that shouldn't have been. Huang suggests generating a serial number only after all tests pass, as a last step. This way, the serial number itself acts as a unique proof that the device can be shipped. You can also use a spreadsheet to keep track of individual units or prototypes and their status. For example, you might track the ID or serial number, status, location, and problem history. This works best for prototypes or a small number of units and can be used for subassemblies or entire assemblies.

NOTE

For Andrew "bunnie" Huang's excellent article on factory testing, see https://www.bunniestudios.com/blog/?p=5450/.

There are also companies that will do test tracking and logging for you. Instrumental.com offers a service that will monitor production lines and give you a picture of every device off the line with proof that it was tested according to your test plan and that it passed.

The ability to debug and update your test jig during production is critical and can sometimes save you a flight. Inevitably, something will go wrong or break, and you'll need to figure out why and how to fix it. Time is very expensive when production is halted, so having your own complete duplicate test jig specifically for development and debugging is really important, especially if you're not in the same country as the production line. If internet is available in the factory, designing the test jig to have remote access can be valuable for working through issues or updating software. Workers can sometimes assist by getting on a video call.

Designing Tests

Designing good tests is difficult. The good news is that you don't have to start from scratch. There are myriad test specifications out there. IEC, FDA, JEDEC, NIST, and other organizations lay out standards for the exact test procedures for many types of products. Some of those standards include:

- MIL-STD-810
- MIL-HDBK-310
- SAE J1211
- IPC-SM-785
- Telcordia GR3108
- IEC 60721-3

Even if your product doesn't fit into exactly one box, you can pick and choose tests from different standards to start out with. The goal is to completely test your device, end to end, in relevant, accurate conditions. Here's a list of the most common factors used to qualify a design and the parameters you may need to test:

Temperature Maximum, minimum, and ramp/dwell times

Humidity Condensing or noncondensing

Corrosion Salt, hydrogen sulfide, chlorine, nitrogen dioxide, and so on

Atmosphere Low pressure, high pressure

Power cycling Number of cycles

Supply voltage Maximum, minimum

Supply current Acceptable range, device modes

EMI/EMC Spectral mask / emission limits, dwell time

ESD Voltage

Mechanical strain Static, cycling, torsion

Mechanical shock G-force, waveform, number of events

Drop test Height, surface to drop on

Random vibration Power spectral density, time, kurtosis

Harmonic vibration G-force, frequency

Ingress protection Dust, water, IP level

NOTE

This list of testing parameters was borrowed from Cheryl Tulkoff, "Test Plan Developing Using Physics of Failure: The DfR Solutions Approach," retrieved from https://www.slideshare.net/slideshow/test-plan-development-using-po-f/38885160.

These are only *potential* tests you might run. You need to choose which tests are relevant based on how and where your product will actually be used. If you're designing an MP3 player, for example, you probably don't need to do cycles in a hydrogen sulfide atmosphere. Then again, maybe you're building a radio for use in mines and hydrogen sulfide might come in contact with your device. The best way to decide what parameters to test your product to is to think about the environmental and stress conditions that the device will undergo. You can calculate these conditions, measure them, or run tests to figure out what the conditions will be. Think about the average conditions as well as the variability of those conditions and the worst cases.

There are a number of tests you can run on consumer devices to check for things like plastic yellowing or degradation due to UV light exposure, as well as household chemical resistance testing. These and other tests are available from ASTM International, a standards organization that covers many different materials tests.

Don't forget that your product needs to survive not only in its normal operating environment but also in its shipping environment. Your product packaging and shipping logistics will play heavily into what the shipping environment contributes to your product

requirements and testing. When most people think of intense shipping environments, they think of military or aerospace applications. For example, some military supplies are dropped in by parachute and need to survive the impact, and equipment used on the space station or on a satellite needs to survive a high-vibration environment on the rocket ride up. Exotic concerns like these notwithstanding, the much more common cause of shipping-related product failures is air freight.

A lot of cargo that rides on planes isn't in a temperature- or pressure-controlled area. This means it can experience rapid changes in temperature (which can cause moisture to condense on the device), rapid changes in pressure (which can *also* cause moisture to condense), and vibration, all at the same time. Cargo planes are typically pressurized to 8,000 to 10,000 feet and will get to this pressure at about 1,000 feet a minute after takeoff. The temperature inside the cargo bay can vary a lot, but it's typically between 49°F and 79°F. If you'll be shipping your product via air cargo, these are requirements you may want to design for. Surviving the shipping process can also be a task that you offload to the packaging of the product.

While you technically need to test only up to the limits that you guarantee to the customer, it's a good idea to test beyond those limits to points where the product may be misused. Customers will judge how well built a product is based on how realistic the test specifications are and how far past the guaranteed operating points the product will survive. At one extreme, the Mars rovers routinely survive years past their guaranteed or goal lifetimes and are seen as some of the best-engineered devices in the world. On the other hand, if your test conditions cover only part of what you know the device will likely experience, your product will be perceived as shoddy. For example, imagine you design a Bluetooth earpiece and specify that it survive a drop test of 2 feet. This isn't a realistic test condition, since people are inevitably going to drop the earpiece from head height as they're putting it on and taking it off. Even though you can correctly say that the device passed your drop test, consumers will still consider it poorly made because you didn't design and test to a realistic specification.

How do you decide which tests to choose for your product? How do you design new tests? Sometimes you don't have to. The purpose of testing is to ensure that your product meets the specifications you designed it to meet, and in many cases those specifications are the same or very close to the specifications that other products use and were tested to. For example, it's unlikely that you're designing your device to work from –200°C to 200°C. A more common temperature range is something like –20°C to 85°C, which many specifications use. Because of that, there are proven test procedures for these specifications.

Even if existing standards don't have test procedures for the exact specification you're using, you can still extend existing test procedures. That said, it's important not to blindly rely on existing tests. You need to consider the real circumstances under which your product will be used and think about how it can fail given those circumstances. Consider how reliable your users will expect it to be. Is correct operation a matter of life and death? What will the user be satisfied with? What will they be disappointed with? You can't wait until you're designing your test procedures to make these decisions. Rather, you need to think about roughly what your test plan will look like when you're writing your product requirements and specifications. Once you've answered these questions, you can start designing detailed test procedures that will ensure you actually meet your requirements.

A well-designed test has clearly defined success and failure criteria. Each test will start with the materials and equipment needed to run the test and describe how to set them up. Next, the test itself is described, including how to run the device under test, what to record, and how long to run the device. Each test should directly verify at least one requirement. The test should have a defined ending, at which point it can be immediately determined whether the test was passed. The passing criteria should be data driven, not subjective. Anyone should be able to run the same test and get the same result.

You'll need to decide the order in which the tests occur and if any of them need to be performed simultaneously. For example, if your device is going to be on a crab-fishing boat in the North Atlantic, a temperature test combined with a salt corrosion test might be a good

idea, since any salt fog on the outside of the device may freeze. If your device will be touching something with a large temperature differential, that can cause condensation to form on your device and possibly puddle around it.

NOTE

For a great resource on combined environmental testing, see NASA Lessons Learned#643, https://llis.nasa.gov/lesson/643.

Make sure the final version of the product you ship out the door has been fully tested. Last-minute changes and tweaks are all too common, and it's easy to let those small changes make it into the final product without being fully tested because of time or budget constraints. Ideally, you can perform complete end-to-end testing with the version that ships, but often this isn't practical. Instead, use your best judgment to identify what new risks there are as a result of the last-minute changes. For example, if you remove an extra screw on the enclosure, you probably don't need to redo any electrical testing, but you'll likely want to redo vibration, drop, or ingress-protection tests.

The Failure Point

While not possible in all designs, it can be valuable to test devices to their failure points. This will not only inform future design iterations but also help you understand what the signs and symptoms of your product failures are. When you get a device sent back for repair or need to understand why devices are failing in the field, having a complete set of point-of-failure test results that have been analyzed to their root cause will be enormously helpful in identifying the problem, fixing it, and preventing it from happening again.

One measurement that may come up during reliability testing is *mean time between failure (MTBF)* or *mean time to failure (MTTF)*. As the names imply, this is the average time that it takes for a device to fail. Getting accurate values for MTBF and MTTF can take a long time because you need to run a lot of tests to get enough data for statistically

significant results. However, knowing these values can give you a helpful insight into how many warranty claims you can expect to get and when you'll start seeing them. If you see *any* failures when testing a small number of devices, this represents a serious problem since the sample size is small.

There are two main ways to determine whether a device will fail prematurely. If your product is likely to fail after a number of cycles rather than after an amount of time, you can test it more rapidly than it would normally be used. This is called *compressed-time testing*. For example, a mechanical latch may be opened only a few times per day, but you can estimate its expected lifespan by actuating the latch hundreds or thousands of times per day. Another shortcut for determining when and how a device will fail is *highly accelerated life testing (HALT)*, also known as *accelerated stress testing* or sometimes *burn-in*, which simulates the device's aging process. This involves stressing the device to higher than its normally rated specifications under the assumption that stressing it in this way will cause the same failures in the same way as a device that is run at its nominal specifications for a much longer time. No one has time to sit around and run a device for 5,000 hours. Instead, the temperature and voltages on a device are both increased, and the device is run for something like 48 hours. It can be a good idea to let HALT testing continue not just for a fixed number of hours but until something fails, particularly if the number of devices under test is small. This will give you more information about what the expected lifetime will be. Table 13-1 lists some common accelerated tests, including varying protocols for simulating standard-life, long-life, and automotive use cases.

Table 13-1: Common Accelerated Life Tests

Test	Standard life	Long life	Automotive
Autoclave (JESD22-A102)	Optional test, 48 hours	Optional test, 96 hours	96 hours
Data-retention bake (JESD22-A117)	504 hours at 150°C	1,008 hours at 150°C	1,008 hours at 150°C

Test	Standard life	Long life	Automotive
Electrostatic discharge, charged-device model (JESD22-C101)	+/– 500 V (or classification)	+/– 500 V (or classification)	+/– 750 V for corner pins, +/– 500 V for all other pins (or classification)
Electrostatic discharge, human-body model (ANSI/ESDA/JEDEC JS-001-2012)	+/– 2,000 V (or classification)	+/– 2,000 V (or classification)	+/– 2,000 V (or classification)
Electrostatic discharge, machine model (JESD22-A115)	Optional test, +/– 200 V (or classification)	Optional test, +/– 200 V (or classification)	+/– 200 V (or classification)
Highly accelerated stress testing (JESD22-A110)	48 hours at 130°C, 132 hours at 110°C	96 hours at 130°C, 264 hours at 110°C	96 hours at 130°C, 264 hours at 110°C
High-temperature operating life test (JESD22-A108)	5-year equivalent use time	10-year equivalent use time	1,008 hours at 125°C or 504 hours at 175°C
High-temperature storage life (JESD22-A103)	Optional test, 504 hours at 150°C or 240 hours at 175°C	Optional test, 1,008 hours at 150°C or 504 hours at 175°C	1,008 hours at 150°C or 504 hours at 175°C
Latch up (JESD78)	+/– 100 mA (or classification)	+/– 100 mA (or classification)	+/– 100 mA

Test	Standard life	Long life	Automotive
Temperature cycling (JESD22-A104)	200 cycles from –65°C to 150°C, or equivalent per JESD94	500 cycles from –65°C to 150°C, or equivalent per JED94	500 cycles from –65°C to 150°C, or 1,000 cycles from –50°C to 150°C

It may seem unbelievable that HALT will cause the same failures as running the device normally for a very long time, but the research seems to show that this is mostly the case. Neither compressed-time testing nor HALT is a perfect measure of reliability, but the industry generally considers them to be valid methods of testing the lifespan of your device without actually waiting for its entire lifespan.

Electrostatic Discharge

Electrostatic discharge (ESD) testing is typically performed on one of three models: the human-body model, the machine model, or the charged-device model. The *human-body model* represents ESD caused by charged-up humans touching your device. The *machine model* represents ESD caused by conveyors or other moving machinery parts. The *charged-device model* represents what happens when your device gets charged up and is then put in contact with a material of a much lower voltage.

Some people are very concerned about touching a part and killing it with ESD. In fact, a 2002 study showed that only 0.1 percent of documented ESD damage was caused by a human discharging into a component, while 99.9 percent of the cases had damage due to the charged-device model. (I can confirm this anecdotally: At times I've been quite cavalier in not using ESD protection when soldering or handling components, and I've *never* killed a part due to ESD.) This doesn't mean you shouldn't use any ESD protection when handling parts, and the study doesn't speak for every environment, but what you should take away is that one of the biggest risks of incurring ESD

damage is actually from your components getting charged up and then discharging on another object. Make sure all of your lab equipment is well grounded, including your soldering iron and your work surface. ESD mats are one way to make sure your work surface is ESD safe.

NOTE

For the details of the study, see Roger J. Pierce, "The Most Common Causes of ESD Damage," Evaluation Engineering, *November 2002.*

Humidity

Humidity testing can be broken up into two primary types of tests: condensing and noncondensing. In a *condensing humidity test*, the device under test is at a lower temperature than the dew point inside the test chamber. When this happens, the moisture in the air condenses into droplets on the device. A *noncondensing humidity test* is run at lower chamber temperatures and slightly lower humidity so that the water in the air doesn't condense.

It's possible that your device under test will rarely or never have condensation on it because it will never get cooler than the dew point of its environment. For example, the device might get hot after it powers on. In that case, condensing humidity tests may not be necessary. However, before you mark it as immune to condensation, think about the entire temperature range it will be subjected to and whether turning the device on and off at the right time and temperature could lead to condensation, even if only for a few minutes. You should also think about whether condensation can drip onto your device. Maybe the device itself runs warm enough that it will always be above the dew point, but a cable or antenna above it doesn't get warm and therefore allows condensation to form. Also, conformal coatings on their own aren't necessarily enough to protect against condensation forming on a PCB, so you don't get to skip humidity testing just because you're using a conformal coating.

The point of any humidity test is to detect signs of water ingress into areas where it shouldn't be (which can cause shorts), as well as electrochemical migration. Copper and other metals can form dendrites that can cause shorts. Dendrites form when metal ions dissolve in water (like condensation) and start redepositing as crystals. These crystals grow between areas with a potential difference. When they touch, they begin conducting and cause a short. Ion migration happens very quickly in the presence of water, and that's why it's important to run humidity tests to try to identify and prevent it. High voltages can also catalyze the reaction and cause faster-than-normal growth. There are some subtleties to the different mechanisms and types of electrochemical migration, but you may also hear these kinds of effects referred to as *tin whiskers* or *tarnish creeping*. See Chapter 7 for more information on tin whiskers.

There are several different standards that describe humidity tests. Figure 13-2 shows an example MIL-STD temperature-humidity test curve. Each full cycle takes 24 hours, and during that period the temperature is quickly ramped, held, slowly transitioned, and held again. You can use this profile for your humidity tests directly or use it as a starting point to create one for your application.

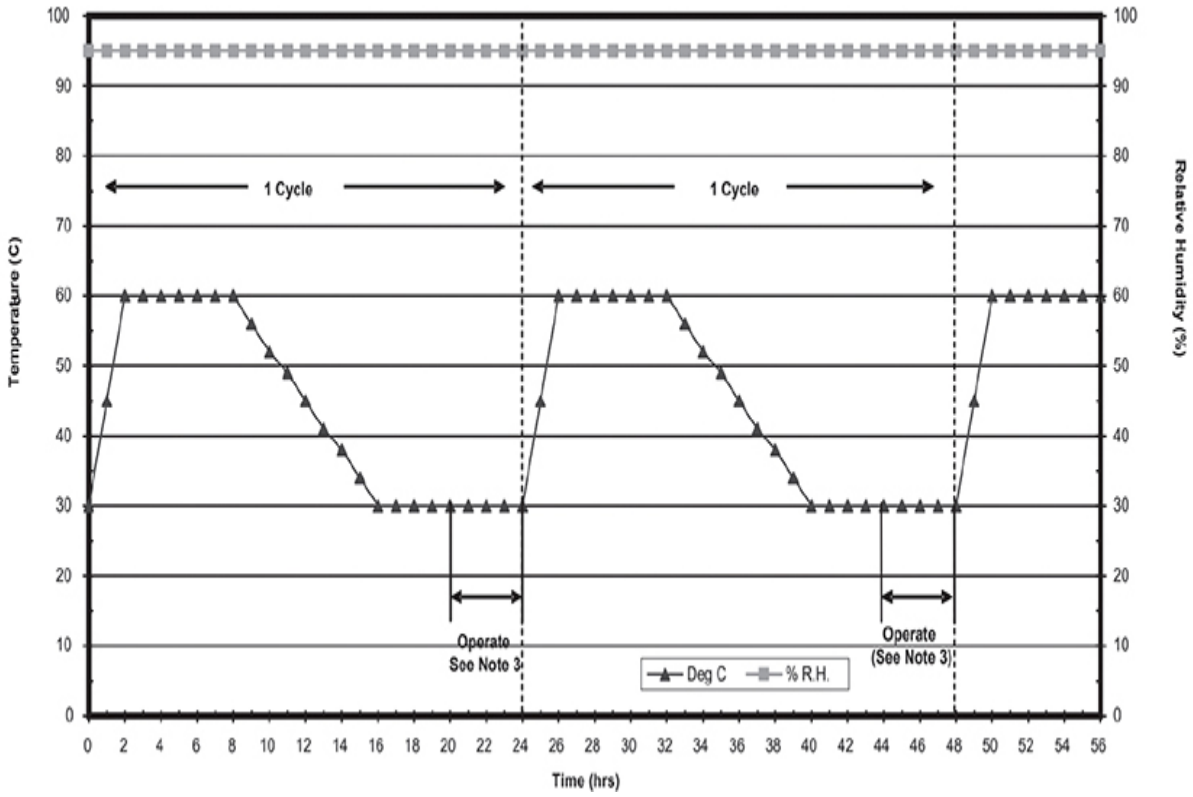


Figure 13-2: An example humidity testing curve from MIL-STD-810g. See the standard for details and additional test curves.

Table 13-2 lists several other standards for humidity testing.

Table 13-2: Humidity Testing Standards

Standard	Temperature	Humidity	Duration
IEC 60068-2-3	40°C	93%	500 hours
IEC 61189-5	40°C	93%	72 hours
IPC TM-650, 2.6.3.5	35°C	88%	96 hours
IPC TM-650, 2.6.14.1	40°C	93%	500 hours
IPC TM-650, 2.6.14.1 (alt)	65°C	88%	500 hours

Standard	Temperature	Humidity	Duration
IPC TM-650, 2.6.14.1 (alt)	85°C	85%	500 hours
IPC 476A	65°C	85%	500 hours
GR-78-CORE	35°C	85%	96 hours
GR-78-CORE (alt)	65°C	85%	500 hours
ISO 9455-17	40°C	93%	168 hours

Failures in humidity testing will often happen intermittently, so you need to monitor your test continuously. If you simply check the device's functionality after the test is completed, it may pass but you'd miss the temporary failures that happened during the test. It's recommended that you run at least three test articles through humidity testing, place them in different orientations, and keep the product continuously powered for the entire duration of the test.

A related test that a lot of consumer products advertise nowadays is *ingress protection (IP)* certification. An IP rating has two numbers: The first represents the degree of protection against solid particles, and the second represents the degree of protection against liquid. Here's what the solid particle ingress numbers mean:

0 No protection at all.

1 Protection against any large surface of the body, such as the back of a hand, but no protection against deliberate contact with a body part.

2 Protection against fingers or similar object.

3 Protection against tools, thick wires, and the like.

4 Protection against most wires, slender screws, large ants, and the like.

5 Ingress of dust isn't entirely prevented, but dust must not enter in sufficient quantity to interfere with the satisfactory operation of the equipment.

6 No ingress of dust; complete protection against contact (dust-tight). A vacuum must be applied. This requires a test duration of up to eight hours based on airflow.

And here's how the liquid ingress numbers are determined:

0 No protection at all.

1 Vertically drip water equivalent to 1 mm rainfall per minute for 10 minutes.

2 Tilt the device 15 degrees and vertically drip water equivalent to 3 mm rainfall per minute for 10 minutes.

3 Spray water at an angle up to 60 degrees at 0.7 liters per minute for 5 minutes.

4 Splash water on the device at 10 liters per minute for 5 minutes.

5 Use a 6.3 mm nozzle at a distance of 3 m from any direction to spray 12.5 liters per minute for 3 minutes.

6 Use a 12.5 mm nozzle at a distance of 3 m from any direction to spray 100 liters per minute for 3 minutes.

7 Immerse the device in 1 m of water for 30 minutes.

8 Immerse the device in a water depth specified by the manufacturer for the time specified by the manufacturer.

9 Spray water at a rate of 14–16 liters per minute at a pressure of 8–10 MPa at distance of 0.10–0.15 meters using water that is 80°C.

The full test standard is IEC 60529, which contains all of the details you need to run these tests. You can even trial some of these tests yourself with a garden hose by buying the right nozzle and a digital flow meter that screws onto the end of your hose.

How to Avoid Problems with Tests

To avoid problems during testing, get to know your testing equipment and what it can and can't do. This will help you get the most out of the equipment's features and prevent you from making mistakes that will aggravate your testing. Pay attention to the warnings about the maximum input conditions, and make sure the tool you're using is actually capable of measuring the quantity you're interested in. For example, are the connectors and cables rated for the frequency you're using? Is your probe and its connection to your circuit going to introduce signal integrity problems? Some people mistakenly treat test equipment (especially expensive equipment) as magical devices that will always just show you the signal you want to see in perfect clarity. In reality, the faster or smaller the signal of interest, the more thought you need to put into how you're going to correctly probe it.

You can't expect to be able to successfully test or debug anything unless you're sure of your equipment's accuracy and its use. This is the problem with cheap test equipment: You can't trust it. Buy or borrow high-quality equipment as much as you can. If you can only get your hands on cheap tools, at least try to understand their error. If you have two multimeters, for example, check them against each other. If you have a suspicious bench power supply (most labs I've seen have at least one of these lying around), measure the output on a scope to see how bad the ripple is and how accurate the output is. Cheap power supplies can have voltage overshoot when they power up, and that can damage your circuit. Another common problem of cheap power supplies is that they're not grounded correctly, which is a safety hazard.

Check your test equipment's settings, and make sure everything is calibrated. If someone was using the equipment before you, who knows what they may have changed. For most equipment, calibration just means it's been serviced by the manufacturer recently. For a vector network analyzer (VNA), you'll have to do your own calibration before you begin a test. You'll need to repeat VNA calibration every so often, possibly multiple times per day depending on the changes in temperature in the room and the manufacturer recommendations.

Make sure that none of your equipment has silently failed. Things like probes, cables, and test equipment inputs can easily be damaged without you realizing it. This is especially true if multiple people are sharing the equipment. Cables will often not fail outright but rather degrade, leading to frustrating hours of debugging or testing only to discover that the problem was a bad cable. If you have any evidence that a cable is damaged, throw it away. Don't put it back on the rack for someone else to repeat your ordeal!

Test the insertion loss of your RF test cables periodically, and make sure the loss is what you expect. Moving RF test cables around can weaken them, and in some cases the coax outer conductor braid can shear off from the connector, rendering the cable totally useless. If an RF cable is acting funny, either label it with the S21 or throw it away. RF adapters can also cause problems with measurements, so make sure to check them occasionally, too.

When testing amplifiers, the outputs need to be terminated into a load. Usually this is 50 Ω. If the amplitude of your output is n watts, you need at least an n-watt terminator. Even though terminators are just resistors, keeping a 10-watt signal on a 2-watt resistor for even a few seconds is enough to blow up the terminator. It will probably fail as an open, which will create a huge mismatch, cause a huge voltage standing wave ratio, and very possibly blow the amplifier. It's a good idea to measure your terminators every once in a while, just in case someone has accidentally killed one and didn't realize it. Even if a bad terminator doesn't blow up your amplifier, it can still throw off all of your measurements.

Conclusion

It can be intimidating to release your product into the wild, where strangers will use it and expect it to solve their problem and work every time. A well-designed test campaign is the key to confidence that the product will do what you expect. It's how you know you're done with product development, since testing allows you to prove you can meet all of the product requirements. It's also how you know your product will

survive contact with the real world. All the time you spend designing and executing thoughtful tests will pay itself back in happy customers.

14
TROUBLESHOOTING

Strong troubleshooting skills are among the most useful assets a designer can cultivate. Nothing ever works exactly as planned on the first try, so troubleshooting is a key step in the electronics design process. In this chapter, we'll cover a variety of tips and techniques for effective troubleshooting.

Troubleshooting can be one of the most frustrating aspects of engineering. It almost always takes longer than you wish it did, and you can feel exasperated when it seems like nothing you're trying is working. There's a joke among computer scientists about debugging that applies equally to troubleshooting electronics. It says that the stages of debugging go like this:

1. That can't happen.
2. That doesn't happen on my machine.
3. That shouldn't happen.
4. Oh, I see.
5. How did that ever work?

There's no getting around the fact that troubleshooting is hard. It's something almost everyone learns from experience, not a textbook. Still, there are several strategies you can use to help. Many boil down to a simple principle: In order to successfully solve a problem, you must first identify its true root cause.

Getting to the Root Cause of a Problem

The key to making rapid forward progress in your troubleshooting is the ability to quickly assess the problems you encounter and fix them correctly. This means not just treating the "symptoms" but fixing the root cause of the problem. Fixing the symptoms may buy you a little time (good enough for a hackathon demo or something similar), but in the end it will only make the problem worse. In the case of safety-critical devices, correctly diagnosing and fixing the root cause of the problem is even more important because human lives are on the line.

How close to the root cause you're able to get may depend on the equipment and facilities available to you. For example, discovering that a microvia has caused delamination in an inner layer of a circuit board requires sectioning the board, preparing the section, and looking at it under a microscope. Dedicated failure analysis labs do this kind of thing all the time, but you may not have the tools or skills to do it yourself. In that case, you won't be troubleshooting down to the root cause; you'll be troubleshooting to determine what the problematic component is. In this example, you would take apart your system and eventually isolate the issue to some unknown problem on an inner layer of a particular PCB. You could then either replace the board and hope the same issues doesn't arise on the replacement or send the board to a failure analysis lab and have them figure out the root cause.

How Electronics Fail

To get to the root cause of a problem, it helps to consider generally how electronics fail. Broadly speaking, an electronics failure can be one of two things: a design problem or not a design problem. Design problems are simple: You designed something, but not what you thought.

Somewhere there's a mistake in your understanding of how the system should work, and you either left something out or added in something wrong. If you can find that mistake, you'll know what to do to correct it.

The second kind of failure isn't your fault (directly). These failures are a result of manufacturing problems or environmental factors. The causes of this second kind of failure include the following (taken from M. McMeen, J. Tynes, M. Bixenman, and D. Lober, "Electronic Assembly Warranties Challenge the Industry to Improve Risk Mitigation Test Methods," *2016 Pan Pacific Microelectronics Symposium (Pan Pacific)*, Big Island, HI, 2016, *https://doi.org/10.1109/PanPacific.2016.7428406*:

1. Chemical/contaminant
    (a) Moisture penetration
    (b) Electro-chemical migration
    (c) Pitting / crevice corrosion
    (d) Free ionics not bound or cross linked
2. Electrical
    (a) Electro-migration
    (b) Conductive filament formation
    (c) Thermal degradation
3. Mechanical
    (a) Fatigue
    (b) Creep
    (c) Wear
4. Overstress issues
    (a) Supplier compliance to material specs
    (b) Field loading
    (c) Product durability

One of the most common types of contaminant-related failures that you'll see is the result of moisture ingress. The danger isn't just that water can cause shorting directly but also that moisture can cause other problems. Electro-chemical migration occurs when metal ions get

dissolved in water and voltages cause dendrites to grow. Those dendrites are what actually cause the shorting and subsequent failure. This is why it's important not only to prevent moisture ingress but also to make sure your boards are cleaned to minimize surface ionic contaminants.

Thermal degradation and mechanical wear are more familiar causes of failure. Thermal degradation can be the result of poor thermal management, operation of the device outside of its intended environmental envelope, or neglected maintenance (like cleaning a filter or refilling coolant). Mechanical wear usually happens over long periods of time and may indicate the expected end of the device's lifetime. Both of these conditions can usually be detected early by sensors so the user can be alerted and take action before failure occurs.

Electrical overstress (EOS) is an overvoltage or overcurrent event that typically lasts longer than an ESD event. EOS can cause failures in two ways: dielectric breakdown or thermal damage. An overvoltage condition will break down the dielectric inside a component and conduct through it. This is sometimes called *avalanche breakdown* or *shoot through*. Thermal damage results from overcurrent conditions. A large current through even a small resistance will create heat, and this heat can be enough to damage components. One common example of this is bond wires in an IC melting and creating an open circuit.

Practical Tips

It's good to know what the "atomic" sources of failure are in electronics, but they'll almost never be the first thing you encounter when something breaks. This is why troubleshooting is a process that involves tracing an observed problem down to its root cause, whether it's a design problem or not. It's difficult to give specific troubleshooting advice because everyone's device and environment are different. However, I've spent many, many hours troubleshooting, and have noticed some trends. The following tips capture the thoughts going through my head and the strategies I employ when I sit down with malfunctioning hardware:

Don't assume it's the component manufacturer. If at any point you find yourself wondering if a manufacturer shipped you a faulty component, you're almost certainly wrong. This basically never happens. More likely, something's not hooked up right, or you killed the part. Take another look.

Use a thermal camera. One of the first things you should look for when troubleshooting a problem is accidental short or open circuits, and a thermal camera is a great way to quickly do this. When you apply power, you'll be able to clearly see short circuits caused by a component, since they'll quickly get hotter than everything else. If you have an open circuit somewhere and a part isn't getting power at all, that part will appear colder on the thermal camera while all the parts around it begin to heat up as soon as you apply power. Because modern thermal cameras are so sensitive, even a change of a few degrees that would be difficult to perceive with your finger is very obvious on the screen.

Trust your nose. Your nose is an important but overlooked tool when troubleshooting electronics. There's a very particular smell that damaged parts emit, and with experience you'll be able to identify it almost instantly. Even when a board is inside an assembly or enclosure, putting your nose against it and smelling can tell you if something failed inside. Once you suspect a problem, you can use other tools, like a thermal camera, to figure out exactly which component is at fault.

Use your finger. Your finger can be a very versatile tool: By touching a component, you can add a couple hundred picofarads of capacitance and around 1 megaohm of resistance and inject about 1 microamp of AC current at 50 or 60 Hz. You can also heat or cool a circuit with your finger or apply moisture. And you can also sense temperature (if you're careful).

Don't forget the obvious. When you first start to debug a problem, check for the "obvious" stuff first. Is the device plugged in and turned on? Is the power polarity right? Is it configured correctly? How about your test equipment? Did any wires come

unsoldered? These issues are all quick to check and quick to fix, so check them first before you start tearing the whole device apart looking for subtle bugs.

Check for parts held in reset. If you have problems getting an IC to program or respond to input, make sure the part isn't being held in reset. This can happen either because of a design mistake or because of a short or open on the PCB due to an assembly error. Measure the reset pin of the chip in question. If the chip runs on an external oscillator or clock, measure the oscillator or clock signal with an oscilloscope to make sure it's within the right threshold voltages and doesn't have undershoot or overshoot. If communication buses aren't working, use a logic analyzer to verify that each pin has the right signal on it and that the frames and packet structures are correct. You can also use your scope to make sure the noise and signal integrity of your buses are acceptable.

Consider a field probe. If you're having problems with EMI, EMC, or oscillation of some kind, a field probe is a good way to locate the offending component or area on your PCB. There are two kinds of probes: electric field (E-field) and magnetic field (H-field) probes. They range in cost from a few hundred dollars to thousands. Depending on the frequency of the signals you're trying to measure, you may or may not need to spend a lot of money on probes. You can easily build your own H-field probe for a few dollars with a loop of rigid coax cable, but its frequency response may not be very good at higher frequencies. Some probe kits come with small amplifiers you can use to try to compensate for this. However, if you want to make precise measurements of signals above 1 GHz, you'll need to spend more for a prebuilt and characterized probe set rated for the frequency you're interested in.

Focus on one thing at a time. In general, you should isolate each problem and address it on its own. Don't try to investigate multiple problems at the same time. Change only one variable. Once you start experiencing the problem, start removing parts until the problem goes away. Then, once the problem has stopped, start

adding parts back in until the problem reappears. It's very important that you take a second to think about your experimental setup and whether you're actually changing only one thing at a time.

Hardware or software? If the device you're debugging has both hardware and software, try to first isolate the problem to one or the other. This is easier said than done, and it's possible that the problem is in both, or that there's more than one problem. But think about what tests you can design that will tell you whether you should focus on hardware or software debugging first. If you have one device that's working correctly and one that isn't, try inducing a failure in the functional device to replicate the issues you're seeing on the faulty device. You can also try swapping hardware or firmware between the two to narrow down the source of the problem.

See if you can make it happen on command. Can you write custom firmware to trigger the problem? Can you make the problem worse? Knowing what makes a problem appear doesn't always immediately tell you what the cause is, but it lets you narrow down the list.

Problems don't solve themselves. If a problem seems to have mysteriously solved itself, be very careful. Don't assume that the problem is fixed and will never come back. Instead, take the time to continue troubleshooting. It's possible that a change in the environment, the usage of the product, or the testing process made the problem go away, so try to make it come back. Knowing how to reliably replicate the problem is important so you can understand what conditions are necessary for the problem to occur and so you can test whether you fixed the root cause or just a symptom.

Ask for help. It's invaluable to have a fresh set of eyes look at a problem and try to find possibilities that you've overlooked, especially after you've been debugging something for a long time. Ask people with more experience and with expertise in different fields than you. Sometimes a problem will be obvious to them but invisible to you.

Document your troubleshooting process. You'll have no hope of keeping track of what you've tried and what to try next unless you keep a record of not just your results but also your data and experimental setup. Collect as much data as you can before, during, and after failures. This is important not only for future you but also for when you ask for help from other people. With the right documentation, they'll be able to see what you've done and replicate your results. (You can find more information about keeping a good lab notebook, as well as an example, at the end of Chapter 10. There's also more information about documentation as it pertains to debugging under "Narrative-Based Troubleshooting" on page 291.)

Know the system you're debugging. You can't debug something if you aren't familiar with it. If you designed the whole system yourself, this is easy. It's harder if you designed only part of it or even none of it. Do your homework up front and read the available documentation, talk to the designer(s) if you can, and understand how the device was designed and what design choices were made. Likewise, know the tools you're using to debug and their limitations. The application engineers at the company that made the test equipment you're using are a good resource. They can help you make sure you're set up to accurately take the measurements you intend.

Tests aren't neutral. Consider the effects that adding instrumentation may have on how the system runs. Logic analyzer probes add a bunch of capacitance, ammeters add resistance, and so on. You can try to account for these effects by making sure you can replicate the problem after you add instrumentation. Again, when in doubt, ask the application engineers at your test equipment manufacturer, and they should be able to help.

Quit thinking and look. This is a wonderful piece of advice from the book *Debugging* by David Agans (AMACOM, 2002). Looking is hard, and you're not going to want to do it. You have to set things up, solder wires, replace parts. It's a whole ordeal. You'll sigh when you consider the work you need to do just to get results that may or

may not be helpful. You'll want to sit at your desk and think hard about what the problem could be, talk to people about what the problem could be, or think up new, simpler experiments that you could run instead. On their own, there's nothing wrong with any of these approaches. But don't let them slow you down from actually putting in the work and testing your ideas. That's the only way to get new information.

Troubleshooting Models

It's a lot easier to tackle a complex problem when you have a strategy. The idea of a troubleshooting model is to have an overarching strategy to guide you in your day-to-day work of getting the thing fixed. It serves as a framework to understand what conclusions you can come to based on the data you've collected and what experiments would be most useful to try next. In this section we'll explore some examples of troubleshooting models that I've found helpful.

The Medical Model for Troubleshooting

Electrical engineers can learn a lot from doctors. Doctors solve problems by observing a patient, performing tests, and trying to conclude what the underlying disorder(s) may be. Then they can plan a treatment.

The problems and disorders of living things are quite different from those of a circuit. Still, the principles of medical troubleshooting are readily applicable to electronics. After all, the human body is the most complex machine we know about. Imagine trying to debug a system that complex, and consider the fact that we don't even fully understand how all the parts work! The troubleshooting techniques in this section are good enough for that machine, so it's safe to say they're good enough for whatever devices we design.

Doctors use medical history, physical examination, and diagnostic tests to learn more about the disorder they are attempting to treat. They also frequently consult with specialists to take advantage of experience they might not have themselves. You can use these same techniques

when troubleshooting electronics. If the failure happened in the field, get as much information about the usage history and conditions of use as possible, ideally from the users directly. Carefully examine both the hardware itself and the behavior of the device, and perform tests to help identify the root cause. And don't be afraid to reach out to people who specialize in the suspected problem area. If you don't personally know anyone you can reach out to, contact an application engineer.

Another method doctors use to identify a disorder is something called *differential diagnosis*. This is basically a fancy way of saying "process of elimination." You list everything that's wrong, list all possible causes of those problems, and start trying to rule out the causes. In medicine, this list is prioritized by urgency or danger to the patient. You can follow a similar pattern for electronics if the problem you're seeing can possibly be caused by something that would be unsafe.

If you manage to rule out everything on your list, that means either that you made a mistake and forgot a possible explanation or that this is a never-before-seen problem. Search the internet, ask on forums, and talk to other engineers to try to discover other instances or similar occurrences of the problem you're seeing.

Signs, Symptoms, Etiology

Medicine makes the important distinction between the signs, symptoms, and etiology of a disorder. *Signs* are observations that anyone can make. *Symptoms* are experiences that the patient has that differ from their norms. *Etiology* is the underlying cause of signs and symptoms. By analogy, the signs of an electronics malfunction are things we can observe on the PCB, like a burned-out component or a cut trace. Symptoms would be unexpected or abnormal behaviors of the circuit. And etiology would be the root cause of the problems, like plugging in a battery backward or exceeding the temperature rating of the board.

Pathologists use a set of statements called *Koch's postulates* to help determine if a particular pathogen is causing a set of signs and symptoms. Here's an adapted version of Koch's postulates that you can use to help determine if a certain mechanism is causing a particular problem across multiple units:

- The suspected cause or causes of the problem should appear to be present in all cases known to be defective and not in units that are not defective. Note that this requires full failure analysis of every faulty unit because sometimes signs may present themselves differently in different units, even if they had the same failure cause. For example, an IC can fail catastrophically and be clearly burned, or it can fail internally and look fine from the outside. You may be misled into thinking that you are dealing with two different problems when both of these signs can occur due to the same underlying cause. Failure analysis will prevent this.
- When a problem is fixed, the signs and symptoms of that problem should go away. If the problem starts happening again, the signs and symptoms should come back. If this happens, you likely haven't found the true root cause and merely have been treating symptoms. If different signs and symptoms arise, you may be looking at a new problem with a new underlying cause.
- If you noticed signs or symptoms before a significant failure, or if the signs or symptoms get worse as the suspected underlying cause gets more severe, the suspected cause is more likely to be correct.
- The failure modes inferred from your observations should be consistent with known failure modes of those components.
- The signs and symptoms of the failure should be reproducible if you purposefully create the suspected underlying cause in a known working unit.

Anyone familiar with the original Koch's postulates will know that they've been very heavily adapted here, but the underlying logic for troubleshooting is valuable and works outside of medicine. Some of the postulates may seem obvious or elementary, but they're important to remember. It's very easy to get "lost in the woods" when troubleshooting and start walking in circles or forgetting the basics.

What happens when you have multiple problems stacked on top of each other? Or what if a sign or symptom is just a fluke, a one-time thing? You may be tempted to make intensive observations and try to

correlate behaviors with signs or see if there's a statistical link between behaviors. However, the recommended course of action in the medical world is to perform experiments instead. This is because it can take a long time to gather enough observations for a meaningful analysis, and even then you won't be able to say with absolute certainty if you've correctly identified the cause. By contrast, performing experiments usually means correcting what you think is the underlying cause and seeing what happens. It can also mean changing or perturbing the device to get a better understanding of the "rules" of the symptoms and signs.

When you begin to understand how symptoms work (whether certain symptoms always or never appear together, under what circumstances, and so on), the medical community calls that framework of understanding a *syndrome*. A syndrome can be the result of either a single underlying cause or a series of causes, each leading to the next. For the latter case, there's a troubleshooting technique called "asking five whys." It's a concept used in the Toyota Production System, where you ask "why?" five times when you encounter a problem to try to get to its true root cause. Why did A happen? Because of B. Why did B happen? Because of C. And so on. If you incorrectly identify something as the underlying cause of a problem, but it's really just one link in a chain, the problem will eventually recur. A correct underlying cause isn't a sign or a symptom itself, and it can always explain all other signs and symptoms.

Keep in mind that some underlying causes may have signs and symptoms that are subject to a *promoter*. In other words, the signs or symptoms don't manifest until they're triggered by some event or change of conditions that isn't itself the root cause. For example, if some of your devices start failing whenever the PCB is installed into the enclosure, the underlying cause isn't that the PCB is being installed into an enclosure. Let's say after some more analysis, you eventually determine that the PCB is flexed slightly when it's screwed into the enclosure, and that flex is creating an open circuit on a bad solder joint. This is an example of a problem that already existed but wasn't visible until something increased the expression of the problem (in this case,

screwing it into an enclosure). Flexing the PCB isn't the underlying cause that explains the syndrome here; the bad solder joint is. Flexing the PCB is simply the promoter.

Cause and Effect

A very common problem in medicine is trying to determine whether there's a relationship between a suspected cause and an observed effect. This can be difficult enough for a complex electronic device, but you and I will never design anything as complex as the human body. Studies attempting to find links between behavior or environment and health are published every day, but perhaps the most famous of these was a series of studies linking smoking to lung cancer. Sir Austin Bradford Hill was the statistician and epidemiologist who demonstrated this connection, and he proposed a set of criteria for establishing a link between a cause and an effect. I've adapted them slightly for electrical engineering:

- A small association doesn't mean that there isn't a causal effect, though the larger the association, the more likely that it's causal.
- Consistent findings observed by different persons in different places with different samples strengthens the likelihood of an effect.
- Causation is likely if there's a very specific sign or symptom at a specific location that has no other likely explanation. The more specific the association between a factor and an effect, the bigger the probability of a causal relationship.
- The effect has to occur after the cause (and if there's an expected delay between the cause and expected effect, then the effect must occur after that delay).
- Greater exposure to the underlying cause should generally lead to greater incidence of the effect. However, in some cases, the mere presence of the underlying cause can trigger the effect. In other cases, an inverse proportion is observed: Greater exposure leads to lower incidence.

- A plausible mechanism between cause and effect is helpful (but Hill noted that knowledge of the mechanism may be limited).
- Occasionally, it's possible to appeal to experimental evidence.
- The effect of analogous similar factors may be considered.

Let's look at an example that illustrates these criteria. Say you're analyzing failures of your new product, an outdoor wireless router. You notice that you are getting periodic clusters of complaints from customers that routers have stopped working temporarily but seem to fix themselves after a few days. You suspect that the cause may be cold weather and are trying to determine if that's really true. You see a small association between cold weather and these failures. According to Hill's first criterion, just because the failure doesn't happen every time it gets cold doesn't mean that cold weather can't be causing the problem. You also notice that these failures happen across different batches of devices sent to customers in different parts of the world. According to Hill's second criterion, this makes cold weather more likely to be the cause, since the same problem is observed by different people in different places with different units.

You decide to visit a customer with a failed unit to catch it in the act and notice ice build-up on the antenna. Hill's third criterion tells you that because this is a very specific sign at a specific location, the probability of cold weather being the problem has increased. You can also use Hill's sixth criterion, since there's now a plausible mechanism between cause and effect: Ice build-up detunes the antenna, which causes the link to drop. Your field observations tell you that the failures always happen *after* a winter storm, meaning your suspected cause precedes the effect, as required by Hill's fourth criterion. Likewise, you determine that longer winter storms cause a longer delay until the router starts working again, which fits with Hill's fifth criterion.

You try an experiment in your lab where you place a block of ice in front of the antenna of the router and measure a huge packet loss, appealing to experimental evidence per Hill's seventh criterion. Finally, you know that changing the relative permittivity of the medium that the antenna is mounted against will change the performance of the antenna,

which is an analogous effect that Hill's eighth criterion tells you further raises the probability of your cause being correct.

This was just a simple example to illustrate the ideas of Hill's criteria—real-world scenarios probably won't check every single box quite so neatly. Still, remembering these criteria can really help you find your way from an effect back to a cause when you're immersed in the fog of debugging.

Narrative-Based Troubleshooting

Narrative-based troubleshooting isn't a precise troubleshooting methodology but more of a general practice that can make troubleshooting easier. The idea is that you treat your lab notebook like a story instead of a spreadsheet. Write in it like a journal. Here's an example:

> Today I picked up from yesterday and replaced C12. I thought it might have heat damage because it looked pretty burned out. I didn't have another 12 µF capacitor, so I replaced it with a 10 µF capacitor. Still didn't work. Maybe that filter in front of U4 has too low of a roll-off. I'm going to replace R34 with a 4.7k resistor. Okay, that helped a little.

You'll find yourself making a lot of changes during troubleshooting, and it's very easy to forget what you've already tried. Pretty soon you'll end up going around in circles and losing track of the state of your device, which can make troubleshooting much harder than it has to be. This is why writing everything down in a lab notebook is critical.

The advantage of documenting the troubleshooting process as a story instead of just writing down the test results is that it can be easier to follow what's going on, where you've been, and where you want to go. It's especially useful if you take a prolonged break or go on vacation during the troubleshooting process and come back later. Reading through what you were thinking in narrative form can help illuminate areas you missed and areas you've already covered. It's also much easier for another person to take over if they need to, since *all* of your reasoning is written out instead of just the results.

Scientific Troubleshooting

Scientific troubleshooting is using the scientific method to discover a root cause. There are actually several ways to perform what we call the scientific method. They're all similar and mostly differ in terms of the philosophy that underlies them. This book isn't a study of epistemology, so we'll constrain ourselves to the following set of steps:

1. Use your experience. Consider the problem and try to make sense of it. Gather data and look for previous explanations. If this is a new problem to you, then move to step 2.
2. Form a hypothesis. When nothing else is known, come up with an explanation.
3. Deduce predictions from the hypothesis. If you assume your hypothesis is true, what consequences follow?
4. Test or experiment. Look for evidence that conflicts with your predictions in order to disprove your hypothesis.

Rather than trying to disprove their hypothesis, many people make the mistake of looking for evidence that it's true and overlooking or ignoring evidence to the contrary. You see this most often when you really want a particular hypothesis to be true. Perhaps proving your hypothesis wrong would mean having to spend a lot of money to fix the issue, set your schedule back more than you want, indicate that your development path is wrong, or mean that you have to humble yourself and admit you made a mistake. When your project and/or reputation are on the line, it becomes easier than you might think to try to prove yourself right and ignore contrary evidence without even realizing it. Don't fall into this trap.

Another related mistake is a logical fallacy called *affirming the consequent*. This happens when you take a true statement, like "If the fuse is blown, the device won't power on," and assume that the converse is also true: "If the device won't power on, it's because the fuse is blown." However, the converse isn't necessarily true. In our example, it's true that the fuse must be intact for the device to power on, but there are lots of factors besides a blown fuse that could prevent the device

from turning on—maybe it has a dead power supply, maybe it's not plugged in, or maybe it's been smashed with a hammer.

Affirming the consequent is an especially important fallacy to watch out for when debugging. If someone debugging a device hasn't even seen a particular failure mode, but the symptoms are similar, they may incorrectly assume they know what's causing the problem. This can be exacerbated if they don't have a deep understanding of how the device works. Affirming the consequent can also accidentally happen if you're otherwise mentally pigeon-holed into a single subsystem or discipline. Maybe you're only looking for software problems when the issue is in hardware, or maybe you're thinking through the electrical system when the issue is caused by the mechanical system. This is another good reason why you should talk to people on other teams and get advice from other experts.

Ultimately, it's important to remember that while many people consider the scientific method to be some sort of oracle that produces absolute truth, the reality is that the scientific method is a *process* that can only inch us closer to the truth. That means you can never absolutely verify your hypothesis. There's a famous Einstein quote that encapsulates this: "No amount of experimentation can ever prove me right; a single experiment can prove me wrong."

Conclusion

Troubleshooting techniques always sound easy and obvious when you're reading about them, but they can easily go out the window when you're working on a real problem and feel stumped. Just remember to be logical, write everything down, and stay patient and persistent. You *will* get better at troubleshooting the more you do it, and there is always an explanation for what you're seeing.

A
HOW TO GIVE A DEMO

A bad design with a good presentation is doomed eventually.
A good design with a bad presentation is doomed immediately.
—Akin's Laws of Spacecraft Design

This appendix features strategies and tips for giving an effective demo of your product or technology, which can be a matter of life or death for your company. A successful demo shows off your core technology and effectively communicates your message. Of course, it also helps if the product works when it's supposed to.

Knowing Your Audience

Always make the demos specific to your audience. Don't talk about your product in terms of what *it* can do; talk about it in terms of what *the user* can accomplish with it. There's a big difference between "Our product has 32 gigs of storage" and "Our product has enough storage that you don't need to ever worry about running out of space."

Think about your audience's technical level. Almost no one will admit that they don't understand something during your demo. You can even ask people, "Do you understand?" and they'll lie to your face and say yes, just so they don't feel the slight sense of shame that comes from

admitting they need something explained more simply. If people don't understand what's going on or why your demo matters, they'll leave without getting the point you're trying to make and you won't even realize it. Give background and context for your demonstration.

If you know the audience you're demoing for ahead of time, take the time to learn what they care about, their technical level, and the exact problems they face that you'll be addressing. You can learn this information directly by asking questions or indirectly by researching the people, organization, and market of the audience. Sometimes, however, you'll need to demo to an audience you know nothing about, like at a convention. In that case, before you demo, you'll need to take a minute or two to ask some probing questions to get a critical understanding of your audience. Ask questions that get them talking about the problems they face. As they do this, start thinking about whether they're describing symptoms or a real root problem. Remember, you're not selling them your product, you're selling the outcome that your product will provide. Adjust your demo in real time to their particular needs.

Clarity and relevance aren't the only things you're aiming for in a demo. You're also in a battle for attention, whether you realize it or not. It's you versus the phone in each audience member's pocket. The second they get bored or uninterested, they'll pull out their phone and you're done. There's very little chance of getting them back. Even if you do get them back, they're going to have missed a lot of what you said and did. Be more interesting and compelling than their phone.

Beyond knowing your audience, you also need to be clear about your intentions and the purpose of the demo. Are you trying to get funding? Impress someone? Sell a product to customers? The demo will vary depending on this context. Think about the message you're trying to communicate, and make sure what you're demonstrating conveys and supports your message.

Telling a Story

Giving a good demo requires good storytelling. Experienced storytellers know that facts don't inspire people, stories do. This isn't to say your

story should be untrue, but it should keep the statistics and numbers to a minimum. You can convey the same information that you would in a list of facts by using a story instead, and it will be much more memorable and have a larger impact.

Just like any story, a demo should have a beginning, middle, and end. Plan and script your demo so it has this flow. Who are the heroes? Who's the villain? What's the conflict? The best stories usually follow a similar formula, called *the hero's journey*. There are lots of versions of this formula, but one of the simplest is the one that Pixar uses as a framework for their stories:

1. Once there was a ____. (The hero)
2. Every day they ____. (The hero's ordinary world before the conflict)
3. Until one day ____. (The conflict)
4. Because of that ____.
5. Because of that ____.
6. Until finally ____. (The climax of the story, good triumphs over evil)
7. Ever since then ____. (The moral of the story)

You can use this template to help organize the story you're trying to tell. And don't just tell the story of what's happening in the demo. Tell the back-story. Share your struggles. If your audience doesn't know the story of the product and doesn't understand the conflict and difficulties you've overcome in getting the demo to work, they won't be as impressed as they could be. But more important, sharing your problems and making yourself vulnerable to your audience connects you to them. You want them to feel the pain of trying over and over and failing, and you want them to feel the triumph of finally succeeding. Make it personal. People won't remember exactly what you said anyway. They'll remember how you made them feel. Connect with your audience before you try to communicate the facts about your product.

When you tell your story, be passionate. Don't just explain what the product does, explain why the things it enables make your heart sing. What is your higher mission? What does it all mean?

To really wow people, you need to be unexpected. Have a grand finale or climax to your demo. Keep the audience's expectations low at first so you can have the big reveal at the end. Figure out what your audience cares about and play to that. Start with the "easy" part of the demo and then show more and more functionality, ending with the most impressive thing you have. Make sure the audience understands what's going on behind the scenes and why the demo they're seeing is so amazing. Use analogies to explain what's happening, not numbers and jargon. This is especially important if your product is technically complex. It's really easy, especially as an engineer, to want to dive deep into the technology and implementation—these details are fun to talk about and make you feel smart. However, if you go down that road, you'll lose your audience and it will be almost impossible to get them back. People don't really care about the *how*, they care about the *why*.

To effectively communicate your story, speak slowly, be concise, and use words efficiently. This requires more practice than you think. You can become concise by writing out what you want to say and continually trimming, rewording, and consolidating it. After you know what you're going to say, practice saying it out loud. You can't read what you want to say verbatim, and you can't use notes. You have to memorize it and practice until you sound clear, confident, and natural. At the same time, it's important to sound and be authentic. Focus on practicing rather than just memorizing, since you'll come across as inauthentic if it sounds like you're just reciting a memorized script.

Practical Tips

Here are some practical tips for giving a demo that will help things run smoothly:

Never rely on anything at the venue. For example, don't depend on the local Wi-Fi network to be working and have internet.

If you have to use Bluetooth, pair the devices before the demo. Make sure all batteries are charged. If you're using a projector, test your exact computer or product with the exact projector that you'll use for the demo. The goal in preparing for a demo is to have total control over everything you'll be presenting with.

Do a full dress rehearsal. If you can't perform the demo in the exact room with the exact equipment you'll use, get as close as you can. After the rehearsal, don't change anything; the demo should be perfectly reproducible.

Have a backup. Even after all your preparation, make sure there's a plan B up your sleeve. You can either fall back to a simpler demo or have a video of the demo you made earlier. You don't want to spend 10 minutes fiddling around trying to get things to work. You'll immediately lose the interest of your audience and any momentum you had. Give yourself about 15 seconds to troubleshoot your demo and then fall back to your backup.

Make the product look professional. Demos often involve prototype, unfinished hardware. A nice enclosure goes a long way in making a technology look more real and plausible. You can use a 3D-printed enclosure or a urethane cast. Both of these can be fabricated quickly and fairly cheaply. They'll look much better than a breadboard and pile of wires.

It's okay to fudge a little. While a demo needs to go smoothly and *look* perfect, it doesn't necessarily need to have perfect, flawless tech. There's a famous story about the first iPhone presentation that Steve Jobs gave. It's now iconic (you can find it on YouTube) and shows off the features of a device that blew the competition out of the water. But what no one in the audience knew was that the demo was held together with duct tape, and they cut it very close to the deadline to get everything working. The GSM radio in the prototype phone Jobs was using performed so poorly that they had a miniature cell phone tower behind the stage to make sure it got service. This still wasn't enough to give it full bars of service, so they hard-coded the cell service bars to always show full, even if they

weren't. There were actually three prototypes on the podium that Jobs discreetly switched out at various points when the previous ones failed and had to be rebooted. In the end, of course, they were able to actually deliver on everything the demo promised, which you need to do, too. This is the important distinction between showmanship and fraud: You have to be able to deliver.

Learn from others. Watch other demos to get inspiration for your own. Apple keynotes are great examples of how to design a demo.

Make your demo relevant. Play to your audience and show how you can solve the problems they have. Don't just go down a laundry list of features and show every single one. Tailor the demo to the problem you're solving for this particular audience.

Set up a comparison. Consider showing the audience how much better your device is than the alternative by demoing your product alongside your competitor's product. Here's an example of that: There's a company called Weebly that makes a website builder, and for their demo they rebuilt the websites of the investors they were pitching in five minutes while they watched. This was powerful because these people had spent a lot of time and money on the alternative (paying someone a lot of money and waiting several months), so they had a direct comparison to the "better way." Use comparison and contrast to demonstrate the gulf between your product and the competition. This is a great way to give context to what your demo means and why it's relevant.

Aim for more time with investors, not instant money. If you're pitching investors with your demo, the goal should be to get another meeting, not to get a check written right then. You're trying to buy yourself the chance for a much longer meeting with a full pitch.

Build camaraderie. It's a good sign if investors (or whoever your audience is) start acting like they're on your team and make suggestions to help you. They'll go from asking hard questions to trying to answer those hard questions for you.

Anticipate questions. Think of all the questions you *don't* want people to ask about your product or demo and figure out the answers ahead of time. A perfectly performing demo can be wrecked by a few questions that appear to rattle you and make your audience doubt what they've just seen. Remember that saying "I don't know" or "We don't have that all the way figured out yet" is a perfectly valid and acceptable answer to some questions! No one has everything figured out about a new product, and pretending you can see into the future is often worse than just being honest.

One last tip: If you're giving demos to try to raise money, show the demo *before* you need the money. Fundraising is a whole other topic, but I'll leave you with this oft-quoted line in Silicon Valley: "If you want advice, ask for money. If you want money, ask for advice."

B
RECOMMENDED RESOURCES

This appendix lists companies that can help with various stages of the design process. None of these companies paid me or even asked me to include them here. Either I've used these companies or friends of mine have used them. I'm not endorsing any of them, since I have no formal relationship with any of them and can't keep track of whether they're still good. But this list should at least give you some leads for help in the relevant areas.

Tools

GrabCAD A free online repository of thousands of 3D models, including models of electrical components (*https://grabcad.com*).

Memfault Provides a software development kit for embedded software that handles firmware release management, remote programming and development, error monitoring, and telemetry (*https://memfault.com*).

Octopart Lets you search for electronic components with different distributors and manages your BOM to optimize cost and ensure part availability (*https://octopart.com*).

SnapMagic Search A massive source of verified PCB footprints, schematic symbols, and 3D models (*https://www.snapeda.com*).

Upverter A free, completely in-browser PCB CAD tool with as many features as most desktop-based EDA tools (*https://upverter.com*).

PCB Fabrication and Assembly

AdvancedPCB Formerly known as Advanced Circuits, a PCB manufacturer that offers a "bare-bones" service with reduced features but a one-day turn time (*https://www.advancedpcb.com*).

CircuitHub A turnkey PCB fabricator and assembler. Its quoting process is entirely online and instant, and you can use a slider to see how your cost will change when making a single board versus 10,000 (*https://circuithub.com*).

Fusion PCB Seeed Studio's PCB fabrication service. It's based in Japan and has longer lead times, but its prices are very low (*https://www.seeedstudio.com/fusion_pcb.xhtml*).

Gorilla Circuits A PCB manufacturer and one of the best assemblers I've ever worked with (*https://www.gorillacircuits.com*).

MacroFab A turnkey PCB manufacturer and assembler based in the US (*https://www.macrofab.com*).

OSH Park Offers two-layer, four-layer, and flex PCBs at extremely low costs but with slightly longer lead times (*https://oshpark.com*).

OSH Stencils Makes low-cost solder paste stencils to aid in assembly (*https://www.oshstencils.com/*).

Small Batch Assembly A PCB assembler that trades design flexibility for very low cost (*https://www.smallbatchassembly.com*).

Contract Manufacturers

Aoyagi A medium-sized OEM with one factory in Shenzhen, specializing in IoT and electronic toy products. They offer a quick turnaround, attention to detail, and flexible terms (*http://www.aoyagihk.com.hk*).

Evermuch A Shenzhen- and Hong Kong–based manufacturer that's small but nimble. It has an MFi license, which means it's authorized to manufacture Apple accessories (*https://www.evermuch.com.hk*).

Ryder Industries A medium-sized accessories manufacturer that makes a lot of gaming accessories and small IoT-type products (*https://www.ryderems.com*).

Sanmina A US-based CM with nearly 80 manufacturing sites across the world (*https://www.sanmina.com*).

Tankya A Shenzhen-based cable and accessory maker. It can handle the entire design, tooling, and manufacturing process and is MFi licensed (*https://tankya.com*).

Worthington Assembly A US-based electronics CM in Massachusetts (*https://www.worthingtonassembly.com*).

Part Fabrication

E-Make Great with painted CNC plastic and aluminum parts, with an emphasis on cosmetic finishes (*https://e-make.co*).

Fictiv Provides 3D printing and CNC services with an instant online quoting system (*https://www.fictiv.com*).

Focus Manufacturing A US-based metal fabricator and CNC shop (*https://www.focusmanufacturing.com*).

Gener8 Offers design and engineering services for small-quantity production of up to 10,000 pieces. They also have experience with medical devices (*https://www.gener8.net*).

JunYi Metal and Plastic A Shenzhen-based manufacturer that can do CNC, injection molding, casting, and other fabrication techniques. It's cheap, fast, good quality, and easy to work with (*https://www.junyimould.com*).

MicroConnex Specializes in flex PCBs and high-density designs (*https://microconnex.com*).

Protolabs A large rapid-prototyping company that can make almost anything out of almost any material, in quantities up to 10,000 pieces (*https://www.protolabs.com*).

Rubber Captain A US-based company that makes custom rubber and plastic parts in China (*https://rubbercaptain.com*).

Sculpteo Offers inexpensive selective laser sintering 3D printing services (*https://www.sculpteo.com*).

Shapeways Can 3D print in a wide variety of materials, but lead times tend to be long (*https://www.shapeways.com*).

Specialty Coating Systems Can do conformal coating with perylene to waterproof your electronics (*https://scscoatings.com*).

Star Rapid Does rapid, small-run prototype production. Tends to be more expensive than other options (*https://www.starrapid.com*).

StrongD Model Technology An expert CNC company that ships quickly and works with mainly aluminum and plastics. Located in Shenzhen (*https://www.cncmachinedprototypes.com*).

Trio Labs Can 3D print objects as big as 1 square meter (*https://trio-labs.com*).

Voodoo Manufacturing Can 3D print parts in quantities from 1 to 10,000. Use it to get enclosures and parts in large volumes without paying for expensive injection-molding tooling (*https://www.voodoomfg.com*).

Xometry Does rapid manufacturing in the form of CNC, 3D printing, sheet metal fabrication, injection molding, and urethane casting (*https://www.xometry.com*).

Materials

McMaster-Carr Carries nearly every mechanical thing or raw material you could possibly think of. It also has one of the best-designed websites and user experiences I've ever seen (*https://www.mcmaster.com*).

TAP Plastics Has a huge variety of plastics, as well as lots of molding materials. Great for putting together prototype enclosures (*https://www.tapplastics.com*).

Paperwork

Cognition IP A law office dedicated to IP protection and patents (*https://www.cognitionip.com*).

Enzyme Offers a quality-management system and other tools to help with your FDA submission process (*https://www.enzyme.com*).

Schox Patent Attorneys One of the most respected law firms in Silicon Valley. Its lawyers are very technical (many have engineering degrees) and write great patents (*https://www.schox.com*).

Shipping and Logistics

Drip Capital Can help you finance your imports and exports (*https://www.dripcapital.com*).

Flexport A freight forwarder and customs broker. It can help you get your product from your factory's loading dock to your customers' front doors (*https://www.flexport.com*).

Flowspace Can warehouse your product, deliver it, and perform other logistics tasks like repackaging and labeling (*https://flow.space*).

ImportGenius Allows you to search ocean freight shipping manifests and find the CM of products you admire or those that are similar to yours. Starts at $149 per month (*https://www.importgenius.com*).

ShipBob A shipping fulfillment company that lets you offer two-day shipping (*https://www.shipbob.com*).

Sourcify Organizes your supply chain in one place and can help you find an overseas factory to source your product (*https://sourcify.com*).

Design Services

APROE Does great work building early-stage and mechanically complex prototypes (*https://aproe.com*).

Model Solution A South Korean firm that creates very high-quality, expensive looks-like models that are suitable for photoshoots (*http://www.model-solution.com*).

Radicand Specializes in product design, mechanical and electrical engineering, and embedded systems. They can help you make your product from scratch and are used to working with startups (*https://www.radicand.com*).

Chip Fabrication

BRIDG An IC fabricator that caters to research groups and startups (*https://gobridg.com*).

SiFive Allows you to make custom system-on-a-chip hardware based on RISC-V (*https://www.sifive.com*).

SkyWater Can help you design and fabricate application-specific ICs (*https://www.skywatertechnology.com*).

Component Distributors

DigiKey One of the largest electronics distributors. It's often the first stop for finding and buying electronic components (*https://www.digikey.com*).

Mouser Another very large electronics distributor that carries some things that DigiKey doesn't, like parts from Coilcraft and some test equipment (*https://www.mouser.com*).

RFMW A distributor that specializes in RF components (*https://www.rfmw.com*).

Richardson RFPD Another distributor that specializes in RF components. It has some overlap with RFMW, but it also represents manufacturers that RFMW doesn't (*https://www.richardsonrfpd.com*).

Fulfillment

DCL A major fulfillment center for several large hardware companies (*https://dclcorp.com*).

Floship A B2C fulfillment company out of Hong Kong that especially likes working with crowdfunded projects. It can work with small quantities and is good for fulfillment in Asia and Australia (*https://www.floship.com*).

Fulfillrite A fulfillment company great for US East Coast distribution (*https://www.fulfillrite.com*).

Rush Order Provides customer service, order taking, and global fulfillment, with locations around the world (*https://rushorder.com*).

Shipwire Acts like a software layer on top of lots of fulfillment centers. It has an online tool for estimating shipping costs and is pretty easy to work with. The downsides are that it's more expensive and it's harder for the company to fix problems within the fulfillment center because it's one layer removed (*https://www.shipwire.com*).

Simple Global A smaller fulfillment company that provides great service to young, growing companies (*https://www.simpleglobal.com*).

C
EXAMPLE FABRICATION NOTES

This appendix shows an annotated set of fabrication notes for a design. You can use it as a template for your own fabrication notes. Not all of the fields are necessary, and many fabricators already follow some of the callouts listed, but this gives you an example of a pretty complete and detailed set of notes.

Fabrication Notes

- All dimensions are in mils
- Linear and angular tolerances are 0.01 inches, 0.5°
- Interpret in accordance with DOD-STD-100
- Number of layers = 8
- Fabricate per IPC-ML-6012B, Class 2
- Inspect per IPC-600 latest revision

Material

- Type: 370HR

- Raw Cu weight outer layers (see stackup)
- Raw Cu weight inner signal layers (see stackup)
- Raw Cu weight inner plane layers (see stackup)
- Special material requirements:
  - ☐ High Tg[1]
  - ☐ RoHS[2]
  - ☐ Halogen free[3]

Processing

- All vias and plated through holes to be located within 75 µm of true position
- Annular ring of 125 µm nominal, 100 µm minimum
- Copper thieving not allowed
- Remove all unused pads on artwork on all inner layers
- Final conductor and pad widths to be within 25 µm of artwork originals
- Registration +/– 50 µm
- Etch date code and UL-recognized vendor mark on secondary side[4]
- Fabricator shall test and stamp passed circuits on the secondary side
- Optimize circuit arrangement for best panel yield
- Boards must be routed and free of burrs
- Warp and twist not to exceed 0.010 inch/inch

Impedance

- Use requirements in stackup. If none, then stackup is total thickness and layer structure only

- Noncontrolled traces will be within 20 percent of specified aperture value in artwork
- Fabricator must inform, beforehand, of deviations required to achieve impedance
- Transmission line impedance measurement report required

Cu Plating

- Unless otherwise specified, hole dimensions are for finished holes[5]
- Unless otherwise specified, hole barrel thickness to be 0.001 inches minimum[6]

Pads

- Finish all exposed pads with:[7]
  - ☐ Electroless nickel/immersion gold (ENIG)
  - ☐ Immersion silver
  - ☐ Tin/lead HASL
  - ☐ Lead-free HASL
  - ☐ Hard gold over nickel (30 µm/80 µm)[8]
  - ☐ Organic solderability preserve (OSP)[9]

Solder Mask

- Solder mask per IPC-SM-840, Type A, Class 2[10]
- LPI blue solder mask over bare copper, both sides
- Thickness 25 µm +/– 10 µm
- Solder mask registration to be within +/– 0.003 inches of relevant layer[11]
- No mask to appear on pads, except where specified by design (e.g., solder mask defined pads)

Silkscreen

- Use white nonconductive ink
- No silkscreen to appear on pads

Milling

- All board dimensions to be held within 0.005 inches
- If no dimensions present, use board outline to feed directly into milling processor

Vias

- Via special features:[12]
  - ☐ Via in pad
  - ☐ Plugged vias
  - ☐ Plugged nonconductive vias, plated over flat and flush
  - ☐ Via in pad, plugged, conductive, plated over flat and flush
  - ☐ Tented vias primary side
  - ☐ Tented vias secondary side
  - ☐ Blind vias
  - ☐ Buried vias
  - ☐ None
- (Add notes here about which via features should be applied to which diameter vias.)

Assembly

- Solder:[13]
  - ☐ Leaded assembly
  - ☐ Lead-free assembly
- Flux removal:[14]
  - ☐ Clean assembly

☐ No-clean assembly

1. Check this box if you want the fabricator to know that you need a material with a high glass transition temperature (Tg). If your fabricator doesn't have the exact material you specified in stock, they may use a different material with a very similar relative permittivity and loss tangent for convenience and speed. However, it may not have the same Tg, and calling out that you need a high Tg can keep them from accidentally doing this.

2. RoHS, or Reduction of Hazardous Substances, is a European directive that requires your design not to contain any of 10 hazardous substances, including lead. RoHS is often used interchangeably with "lead-free" even though there are nine other things you also must avoid to be RoHS compliant. This applies to both components and solder.

3. Another environmental requirement. Materials that contain halogens like fluorine, chlorine, and bromine are often used in cable insulation and in some components. Burning these materials releases dangerous gases that are bad for the environment and can make fires in enclosed spaces even more dangerous for people trying to escape.

4. You can specify the location for marks the fabricator puts on your board for spacing or aesthetic reasons.

5. This specifies that a hole that's called out as, for example, 10 mils should be drilled slightly larger than 10 mils so that when it's plated with copper, the diameter shrinks to be a finished size of 10 mils.

6. This is specifying the minimum plating thickness of the inside of a via.

7. Select one from the listed options.

8. This means 30 microns of gold over 80 microns of nickel.

9. OSP is a very thin coating that goes over the entire board to prevent oxidation and ensure that the pads are easy to solder to and aren't weakened by any compounds that may form on the pads between plating and assembly.

10. This is the IPC standard for solder mask application

11. This refers to the alignment of the solder mask layer with respect to the copper layer it's covering.

12. Select the options that apply.

13. Select the option that applies.

14. Select the option that applies.

INDEX

Numbers

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z