Practical AVR Microcontrollers

Games, Gadgets, and Home Automation with the Microcontroller Used in Arduino

Alan Trevennor

Apress®

Practical AVR Microcontrollers

ISBN 978-1-4302-4446-2

ISBN 978-1-4302-4447-9 (eBook)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `www.springeronline.com`.

For information on translations, please e-mail `rights@apress.com`, or visit `www.apress.com`.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at `www.apress.com/bulk-sales`.

Any source code or other supplementary materials referenced by the author in this text is available to readers at `www.apress.com`. For detailed information about how to locate your book's source code, go to `www.apress.com/source-code`.

“To Wendy, who made it all possible.”

Contents at a Glance

Contents

Foreword

The computer-on-a-chip functionality of microcontrollers is nowhere better demonstrated than in the Atmel AVR family. The AVR family offers a huge range of microcontrollers, from very simple devices with limited capabilities to much more complex chips that have numerous built-in features for interconnectivity and interfacing as well as much larger memory capacities—all in a single integrated circuit. Yet, all the AVR family use the same basic execution unit for running programs, which means you can start with the simple members of the AVR family and, when you're ready, graduate to the larger, more complex members without having to relearn how they are programmed.

There are lots of interesting books on AVR microcontrollers around, and like this one, many will tell you how to use your desktop machine as a development platform for making AVR programs. However, most other books focus on AVR programs that flash lights, measure temperatures, make sounds, interface to keypads, and so on. As valuable as those things are (and this book covers many of those areas too) it has always been my hope to find a book that has more of a focus on how to make AVRs actually *do* stuff. I want to make things move or change, to make things *happen* in the real world.

There are books that focus on using microcontrollers to provide the intelligence for robots, which is great, but again, not quite what I had in mind. Maybe I watched too much Thunderbirds and Star Trek as a kid, but it always seemed to me that the potential for microcontrollers goes far beyond merely flashing lights. I always imagined that having some degree of intelligence built into a device that could do useful things, and having that device be able to talk to other intelligent devices—all without human intervention—would result in magic!

Having never found the book I wanted and finding that I now had the space and time to play around with some of my ideas, I set about building a lot of them. Now, I am going to share them in this book.

Even if you don't need or want the gadgets and projects I present here in this book, I'm willing to bet that they will spark some of your own ideas and that you'll be able to use many of the concepts and techniques described here to make your own ideas become realities.

That's one major reason for writing this book, to spread the joy! It has been my key learning in creating this book (and yet more stuff that hasn't made it into this one) that the limitless possibilities of the microcontroller stimulate creativity like you may never have experienced before. Another, bigger, reason is that this is Fun, and that's an intentional capital "F"!

It's also worth saying that there are more and more jobs opening up that involve product and project engineering using embedded microcontrollers: if you look you'll see them going on in every sector; from the automotive industry to the defense industries to the entertainment business, they are all finding a tidal wave of applications for microcontrollers. So, if you're of a mind to seek a career using the technology, nothing you learn from within these pages should be wasted.

Alan Trevennor
North Cornwall
United Kingdom
August 2012.

About the Author

Alan Trevennor Alan originally wanted to work in music radio. However, after getting hooked on digital electronics via a Science of Cambridge MK14 computer kit, Alan Trevennor joined the UK computer industry in 1980 as a hardware engineer, fixing DEC PDP-11 systems.

In the 1980s he wrote hardware-related books about operating systems and Amstrad computers. He progressed to systems engineering and became a key member of DEC's UK Unix support team. He created and taught many training courses and user guides for DEC's Unix-related products, RISC computers, TCP/IP networking, and other subjects. He also contributed technical articles to many magazines.

In the 1990s Alan migrated to being a digital media solutions architect with Compaq and then HP. From then until he left HP in 2009, Alan worked on digital media technical solutions and business consultancies. He worked for customers as diverse as the BBC, Reuters, Allied Domecq Leisure, BT, Music Choice, The National Trust, RBS, Glaxo, Virgin Radio, and Nokia. Coming full circle, he later spent a great deal of time in music radio stations as part of a team working on a joint HP/Nokia project—Visual Radio. During an incredibly varied career Alan has created numerous technical solutions (some using AVR microcontrollers) as well as large amounts of user training materials and documentation.

Alan now lives in Cornwall, UK, with his wife and son. He runs a "hobby" business part time and works full time as a technical author for Microtest, a creator and supplier of advanced medical software—based in Cornwall.

About the Technical Reviewer

Cliff Wootton Cliff Wootton is a former Interactive TV systems architect at BBC News.

The "News Loops" service developed there was nominated for a BAFTA and won a Royal Television Society Award for Technical Innovation. He is an invited speaker on pre-processing for video compression at the Apple WWDC conference. He also taught postgraduate MA students about real-world computing, multimedia, video compression, metadata, and researching the deployment of next-generation interactive TV systems based on open standards.

He is currently working on R&D projects investigating new interactive TV technologies, involved with MPEG standards working groups, writing more books on the topic, and speaking at conferences when not lecturing on multimedia at the University of the Arts in London.

Acknowledgments

This book is the product of so many overlapping things that I hesitate to enumerate them all. However, I also hesitate to water it down to basics with an “everyone who knows me” type of generality. So, let me try to steer a middle course.

Special thanks to Wendy for her patience while life went on hold during the writing of this book, and also for letting me use our home to try out my ideas—and being so encouraging about them. Special thanks also to Laurence for his technically appreciative feedback, ideas, and inputs.

To the team at Apress—thank you. To Michelle Lowman who kicked it all off, to Jill Balzano who had the unenviable job of bringing it all together to a tight timescale. To Matthew Moodie who—as technical editor—contributed so much to the structure and content of the book. To Cliff Wootton who as technical reviewer contributed greatly to making the book more understandable and generously contributed ideas of his own for me to adopt. Also, thanks to Lori Jacobs who—as copy editor—made the text much more readable and understandable by spotting mistakes and inconsistencies; things that I would never have spotted if I’d looked at the text for a million years!

In the more general sense, I need to thank Reece Fitzhardy who—all those years ago—freely gave of his knowledge and time to get me to the point where the TTL data book no longer looked like gibberish to me! I thank Alf-Egil Bogen and Vegard Wollan, originators of the AVR architecture, and Atmel who gave it wings!

I thank the folks at Maplin UK, Sparkfun US, and Aaron and the support team at Pololu who have given me help and information whenever I asked for it.

I thank the many thousands of people—but of course most especially the core team—who created something truly remarkable and revolutionary in the Arduino hardware and software.

Also . . . but I’d better stop there. Sorry if I forgot anyone important!

Introduction

The microcontroller unit (MCU) is the ultimate electronics tinker-toy, and in this book you're going to see how to tinker away with it to your heart's delight! My intended audience for this book is those who like to learn hands-on. Learning by doing and seeing has always been my preferred way to learn: If it's yours too, let's take the ride together. For those who like to understand the "why" first of all, the book also includes some background material that explains why using microcontrollers in everyday situations can be such a powerful concept.

My only assumption is that you have some very basic knowledge of digital electronics. But, if that's *not* you, don't worry! There are some appendixes that will give you the start that you need—and the book's web site (and various references through the text of the book) also point you to some valuable AVR MCU-related online resources.

MCU Basics

I'll start with a summary of the absolute basics, just in case you're new in MCU town, if you're not, feel free to skip to the next section. A microcontroller is truly a "computer on a chip."

For straightforward applications such as making LEDs flash, or driving a simple clock display, it's likely that you would only need just one MCU (Microcontroller Unit) chip. For more complex applications (such as those in some of the project chapters of this book) you often need to add helper chips, but the MCU still does all the brainwork.

There are dozens of different microcontroller types on the market (PIC/PICAXE, Intel, ARM, Philips/NXP, Toshiba, Panasonic, and many more) and they all have strengths and weaknesses.

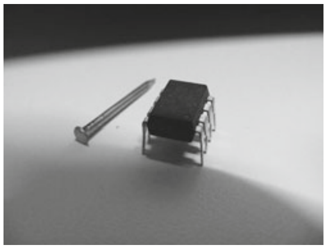
The AVR[1] family of MCUs from Atmel Corporation has become one of the most available and capable general-purpose MCU product sets—and via platforms like the Arduino (more of which later on) has reached a market prominence in the low-cost MCU world. AVR also compares favorably on cost with other low or mid-range microcontroller families.

Microcontrollers evolved partially from the digital memory chips industry and partly from the simpler microprocessors that they have now largely displaced for new designs. We'll be looking at the evolution of the microcontroller in more detail in the first section of this book.

Every AVR MCU consists of a processor core, some programmable flash memory, and some RAM. It will also have on-chip extras, such as input/output (I/O) ports, timers, serial communications ports, analog to digital convertors, and maybe even a USB port.

All chips in the AVR range have the basic processor core and memory, but as you go up the range of products they include more and more of the extras (and bigger and bigger on-chip memory capacities). Using the simplest of AVR MCU chips (a small eight-pin device costing no more than a dollar; see the photo) you can easily make an LED flasher or other simple circuits.

[1]Weirdly, nobody at Atmel wants to tell what the initials AVR stand for. In fact, the guys who invented AVR, Alf Egil Bogen and Vegard Wollan, tease anyone who asks the question! Was it a combination of their initials? No, they say. They even made a teasing video. Search for "The Story of AVR" on youtube to see it.



I mentioned Arduino previously. Arduino is a packaged MCU system that uses an AVR chip at its heart but provides extra facilities such as bringing all the I/O pins of the MCU out to convenient connectors, providing voltage regulation, and so on. We now live in a world where a majority of people are used to using a computer at a high level, using a Windows, Mac, or Linux machine. However, the essential aim of Arduino is to make it easy for non-techs and beginners to try out low-level computing and computer programming for the first time. Low-level computing may use essentially the same technology as your desktop machine, but it's a very different beast.

Arduino is a superb platform, and the software development environment that comes with it is also excellent. However, an Arduino board will cost you between three and four times as much as just an AVR chip, and very few individual projects use all the features of an Arduino board.

So, in many projects it can be beneficial to use just a stand-alone AVR chip with a minimum of external components, and that will be our concentration in this book. As a subsidiary benefit, you will also be more likely to gain a deeper understanding of the AVR from using it outside a packaged hardware environment such as Arduino.

About Our MCU Setup

For many readers, this book will contain a lot of new stuff. To make it easier to assimilate, I have elected to use the Arduino software development environment throughout the book. Arduino's development software (which is 100% free for anyone to download and use) runs on Windows, Linux, and Mac OS X: For the most part, the Arduino software looks and feels the same on them all. So, using Arduino's development environment has the added benefit that you won't be skipping great chunks of the book that don't apply to the machine you are using. Arduino's programming language is very easy to use—another benefit if you're new to all this.

Since this is not primarily a book about programming, we will only be going beneath the covers of the Arduino software when we really have to; that won't be very often, but it will be fully explained when we do. So, although we'll be using the Arduino software, there won't be an Arduino board in sight! We'll use a low-cost AVR programmer board and AVR chips—often we'll be using just the AVR chip on its own.

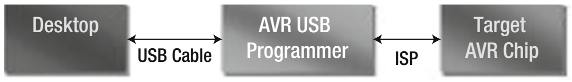
Note Atmel has an excellent (and free) development package of its own, "AVR Studio," which lets you program in the C language, or in AVR native assembler.[2] But at the time of writing it's only available on Windows PCs (XP or later)—inexplicably, there is no Mac or Linux version. So, we don't use it in this book.

---

[2]Assembly language programming is a very low level way of programming, requiring far more knowledge of the intricate details of the chip you are programming and its characteristics. Assembly language programs can often run a little faster on a given processor, but they take very much longer to write and debug than higher-level programming methods like the ones we use in this book.

Putting AVRs to Work

While an AVR is a single-chip computer, it doesn't have anything approaching the capability of your desktop machine—which costs many tens of times more. Therefore, it makes sense to use the greater capabilities, resources, and power of your desktop computer to create the software that an AVR needs and then to download that software into the AVR chip. The following diagram overviews how this works.



On your desktop machine, you install an AVR development environment (all free) which lets you create and compile your AVR software. An AVR programmer (several available) simply connects via USB to your desktop and uses a technique called in-system programming (ISP) to connect to your AVR chip and upload the software you have created into it. I'll go into much more detail and provide a shopping list in Chapter 2.

■ **Note** In this book we'll be using your AVR plugged into a specially set up breadboard (see Appendix C for a basic tutorial on breadboards). However, in other approaches the AVR could be plugged into a circuit board or even a full-scale AVR programmer product.

So, the preceding diagram represents the development environment we will be using throughout this book. It's important to understand that because the AVR family uses erasable and reusable on-chip memory, you can reprogram AVR chips tens of thousands of times if you need to, which means you can keep modifying your program until it's exactly how you want it. Once you have your software exactly right, your AVR chip can be detached forever from the programmer and can go off to have a life of its own in a dedicated application.

In this book we'll be looking at practical examples of how to use useful project elements (such as motors, solenoids, and sensors of various kinds) and software concepts. Then, we'll be making a set of project applications for AVR chips. After you've seen the descriptions and built some of these project applications for yourself, I'm willing to bet that your own application ideas will come thick and fast. It seems that AVRs (and MCUs in general) have that effect on creative minds!

Book Structure

This book is split into two major sections, each of which is further subdivided into smaller sections.

Part 1: Basics

Part 1 deals with the background and the basics. You may already know a lot of this stuff, or you may just be itching to get started with the practical side of things, so feel free to skip any sections that lie outside your area of interest or experience.

We all learn in different ways, and a lot of the stuff in this section is intended for those people who learn better by first understanding "why" things are valuable and "why" one way of doing things is better than another. If you're a "how" learner, you'll probably want to just skim through some bits of Part 1 which deal with history and theory and get onto the more practical sections. If you're going to do that, though, please be sure not to skip the section on setting up your development environment; we'll all need to look at that!

All through Part 1, we will be gently introducing programming, showing you how to program the AVR with some minimal programs that are fully explained so that if you've never programmed before, you'll get the introduction you need.

Part 2: The Projects

Part 2 of this book is all about specific projects using AVRs. These are projects you can build or adapt to your own needs. This section of the book covers a mix of digital electronics, a little lightweight "making" for the controlled mechanisms, software details, and, of course, lots about AVR microcontroller applications.

For each project we'll look at the design of the hardware and any mechanisms needed, discussing any trade-offs and possible adaptations or alternative uses for it. We'll overview the software for the project, detailing any tricky parts of it. The fully commented software will all be available for download from the book's web site: `http://www.apress.com/9781430244462`.

Photographs and diagrams are used to give you as much detail as possible on each project as built, so that you can build one for yourself if you want to or adapt it to your own needs when you make your own version of it.

Following is a list of the projects:

- Chapter 8: "Good Evening, Mr. Bond: Your Secret Panel." Shows how to build a sliding panel mechanism and control it with an AVR. What's behind the panel? Well, wait and see, but I bet you'll soon have your own ideas about what you want behind *your* secret panel, and what secret way you want to be able to open it!
- Chapter 9: "Crazy Beams—Exercise Your Pet!" Cats (and dogs too) are fascinated by moving beams of light, and they get great exercise chasing them around the room. This project gives them all the beams they could ever want to chase, and it never gets tired of playing the game with them!
- Chapter 10: "WordDune: How Much Do You Really See?" We all like to think we could find a needle in a sand dune. Can you find words in a sea of letters? It starts easy, but it gets harder as it goes on.
- Chapter 11: "The Lighting Waterfall: Light the Way—Ever So Prettily!" Don't just "plip" those lights on in that long thin walkway, let's do it with some style!
- Chapter 12: "Moving to Mesmerize":
  - Moiré wheel: Put a light behind a spinning wheel and watch the magic!
  - Animation projector: Flashing LEDs can make shadow magic.
  - Duck shooting game: All the fun of the fair!
- Chapter 13: "Smart Home Enablers." We examine just why the "home of the future" has been so very long in arriving! We look at some get-started foundational projects and ideas that can help make yours a "smarter home."

Appendixes

Finally, we have a number of reference appendixes. These are intended for those "wazzat?" moments, when you encounter an unfamiliar term, technique, or concept. To save time and confusion, readers who are completely new to a subject area might want to read one or more of these before starting on the projects.

- Appendix A: Common Components: Some basics about resistors, capacitors, diodes, LEDs, and integrated circuits (chips).

- Appendix B: "A Digital Electronics Primer." New to the world of digital electronics? Never fear, this appendix is just for you. It won't make you into an overnight digital wizard, but it should give you just enough to get started.
- Appendix C: "Breadboards." What are they, what are they for, how do you use them and why are they so darn useful?
- Appendix D: "Serial Communications." Often puzzling to newcomers, serial communications is a must-understand technology for realizing the full benefits of connected MCU projects.

Where Do We Go from Here?

It's essential that you read Chapter 2 in order to set up your AVR development system. However, after that, if you feel that you already know enough it's not essential that you following any particular reading order: if one subject area appeals to you most, by all means go there first if you feel you already have the knowledge (or are happy to refer back to previously skipped sections).

Make maximum use of the detailed keyword index if you come across the unfamiliar—and don't forget the appendixes, which are there for your reference.

Coming Up Next

Part 1: Chapter 1: "A Brief History of Microcontrollers"—the computer industry takes a RISC.