Preface

This book provides a ‘hardware-free’ introduction to embedded software for people who:

- Already know how to write software for ‘desktop’ computer systems.
- Are familiar with a C-based language (Java, C++ or C).
- Want to learn how C is used in practical embedded systems.

The remainder of this preface attempts to answer some questions which prospective readers may have about the contents.

I What is an embedded system?

As far as this book is concerned:

An embedded system is an application that contains at least one programmable computer (typically in the form of a microcontroller, a microprocessor or digital signal processor chip) and which is used by individuals who are, in the main, unaware that the system is computer-based.

This type of embedded system is all around us. Use of embedded processors in passenger cars, mobile phones, medical equipment, aerospace systems and defence systems is widespread, and even everyday domestic appliances such as dishwashers, televisions, washing machines and video recorders now include at least one such device.

II What type of processor is discussed?

This book focuses on the embedded systems based on the 8051 family of microcontrollers. Prices for 8051 devices start at less than $1.00 (US). At this price, you get a performance of around 1 million instructions per second, and 256 **bytes** (not megabytes!) of on-chip RAM. The 8051’s profile (price, performance, available memory) matches the needs of many embedded systems very well. As a result, the

8051 architecture – originally developed by Intel – is now implemented in more than 400 chips; these are produced by a diverse range of companies including Philips, Infineon, Atmel and Dallas. Sales of this vast family are estimated to have the largest share (around 60%) of the microcontroller market as a whole, and to make up more than 50% of the 8-bit microcontroller market. Versions of the 8051 are currently used in a long list of embedded products, from automotive systems to children's toys.

The low cost, huge range, easy availability and widespread use of the 8051 family makes it an excellent platform for developing embedded systems: these same factors also make it an ideal platform for *learning* about embedded systems. Whether you will subsequently use 8-, 16- or 32-bit embedded processors, learning to work within the performance and memory limits of devices such as the 8051 is a crucial requirement in the cost-conscious embedded market. You simply cannot acquire these skills by developing code for a Pentium (or similar) processor.

III Which operating system is used?

The 256 bytes of memory in the 8051 are – of course – insufficient to support any version of Windows, Linux or similar desktop operating systems. Instead, we will describe how to create your own simple 'embedded operating system' (see Chapter 7). This 'do-it-yourself' approach is typical in small embedded applications, where the memory requirements and expense of a desktop operating system (like Windows or Linux) or of a so-called 'real-time operating system' simply cannot be justified. However, the approach is also in widespread use in large embedded systems (for example, aerospace applications or X-by-wire systems in the automotive industry), where conventional operating systems are generally considered to be too unpredictable.

Learning to work on a 'naked' processor and create your own operating system are key requirements for software developers wishing to work with embedded systems.

IV What type of system is discussed?

This book presents a number of examples adapted from working embedded systems. These include:

- A remotely-controlled robot.
- A traffic-light sequencer.
- A system for monitoring liquid flow rates.
- A controller for a domestic washing machine.

- An animatronic dinosaur.
- A general-purpose data acquisition system.

These and other examples are used to illustrate key software architectures that are in widespread use in embedded designs; the examples may be adapted and extended to match the needs of your own applications.

The book concludes with a final case study: this brings together all of the features discussed in earlier chapters in order to create an intruder alarm system. This case study includes the following key components:

- A suitable embedded operating system.
- A multi-state system framework.
- Software to process the inputs from door and window sensors.
- A simple 'keypad' library to process passwords entered by the user.
- Software to control external port pins (to activate the external bell).
- An 'RS-232' library to assist with debugging.

V Do I need a degree in electronics in order to use this book?

Please consider the following statement:

> *'I'd like to learn about embedded software, but I don't know enough about electronics.'*

This is a concern which is commonly expressed by desktop programmers who – if they ever learned anything about electronics at school, college or university – have probably forgotten it.

If you don't know the difference between a MOSFET and a BJT, or even the difference between a resistor and a capacitor, please relax. **You don't need to have any knowledge of electronics in order to make full use of this book**. Neither will you need a soldering iron, breadboard or any electronic components. In short, this book is (99%) hardware free.

To write software for the 8051 devices considered in this book, we will use an industry-standard (Keil) compiler. To test this software, we will use a hardware simulator. Copies of both compiler tools and the simulator are included on the enclosed CD. Using these tools, all of the examples in the book may be run, modified and recompiled and tested, using a standard Windows PC.

This approach allows experienced desktop programmers to quickly understand the key features of embedded systems before they need to 'get their hands dirty' and build some hardware.

VI What's on the CD?

In addition to the Keil compiler and hardware simulator (discussed in the previous section), the CD also includes source code files for all the examples and the case study: this code is in the 'C' programming language and is compatible with the Keil compiler.

The CD also contains useful information about the 8051 microcontroller family, including a large number of relevant data sheets and application notes.

VII What's the link between this book and your other 8051 book (*Patterns for Time-Triggered Embedded Systems*)?

Embedded C provides an introduction to the use of C in embedded projects. If you want to learn more about embedded systems after you finish this book, then *Patterns for Time-Triggered Embedded Systems* (PTTES) may be of interest.[1]

PTTES is a large (1000-page) book which includes a comprehensive set of 'design patterns' to support the development of embedded systems based on the 8051 family of microcontrollers. In total, details of more than 70 useful patterns are provided, complete with guidelines to help you apply these techniques in your own projects: full source code for all of the patterns is included on the PTTES CD.

The book includes: patterns for embedded operating systems (for both single-processor and multi-processor applications); patterns for user-interface designs using switches, keypads, LED and liquid crystal displays; patterns for PID control; patterns for PWM; patterns for analogue-to-digital and digital-to-analogue conversion; patterns for RS-232, RS-485, CAN, SPI and I^2C serial networks; hardware patterns describing reset, oscillator and memory circuits.

VIII Is the code 'free ware'?

The code included in this book took many years to produce. It is not 'free ware', and is subject to some simple copyright restrictions. These are as follows:

- If you have purchased a copy of this book, you are entitled to use the code listed in the text (and included on the CD) in your projects, should you choose to do so. If you use the code in this way, then no run-time royalties are due.

1. Pont, M.J. (2001) *Patterns for time-triggered embedded systems: Building reliable applications with the 8051 family of microcontroller*, Addison-Wesley / ACM Press.

- If you are using the code in a company, and (for example) ten people are using the code, the company should own ten copies of this book.
- If you are teaching in a university or college, you may freely distribute this code to your students without requiring a licence, as long as the code is used for teaching purposes and no commercial application is involved. Please note that teaching (by university or college staff, or anyone else) of 'short courses' for industry or for purposes of 'continuing professional development' does **not** fall into this category: if in doubt, please contact me for clarification.[2]
- You may not, **under any circumstances**, publish any of the source code included in the book or on the CD, in any form or by any means, without explicit written authorization from me. If you wish to publish limited code fragments then, in most circumstances, I will grant this permission, subject only to an appropriate acknowledgment accompanying the published material. If you wish to publish more substantial code listings, then payment of a fee may be required. Please contact me for further details.

IX How should this book be read?

This short book is intended to be read from cover to cover.

Access to a Windows PC while reading will be useful in later chapters, as this will allow you to try out the examples for yourself: however, this is not essential.

X What about bug reports and code updates?

There is fair amount of code involved in this project, both in the book itself and on the associated CD. I have personally tested all of the code that appears here. Nonetheless, errors can creep in.

If you think you have found a bug, please send me an e-mail (the address is at the end of this preface), and I will do my best to help.

XI What about other reader comments?

I began my first embedded project in 1986. When writing *Embedded C*, I wanted to, try and provide the kind of information that I needed (but could not find) at that time.

2. I can be contacted either by post (via the publishers, please), or much more efficiently by e-mail at the address given at the end of this preface.

I would appreciate your comments and feedback. For example, should the book be longer? Shorter? What other areas should I cover? What should I miss out? Would you like to see a future edition focusing on a different family of microcontrollers? If so, which one?

To ensure that any future editions continue to provide the information you need, I would be delighted to hear of your experiences (good or bad) using the book.

XII Credit where credit is due

The publication of this book would not have been possible without the help and support of a number of people.

In particular, I would like to thank:

- The 'Electronic and Software Engineering' students at the University of Leicester who have provided useful feedback on this material as they attended my introductory courses in embedded systems in recent years.
- Simon Plumtree at Pearson Education, who responded positively to my suggestion that this material was suitable for wider publication.
- Karen Sellwood at Pearson, who helped to keep the project on the rails.
- Reinhard Keil and his colleagues, for reviewing the first draft of this book and – again – providing the core of the CD.
- Jim Cooling, for his review of the first draft of this book.
- Chris Stephens, for his review of the first draft of this book.
- Penelope Allport for managing the project.
- Sara Barnes for copy editing; Claire Brodmann for the design; Barbara Archer for proof reading and David Worthington for the index.
- Barbara and Gordon Pont for proof reading.
- Sarah, for convincing me that 'No More Shall We Part' was worth listening to again.

Michael J. Pont
Great Dalby, February 2002
Michael.Pont@tesco.net