Foreword

Embedded microcontrollers are everywhere today. In the average household you will find them far beyond the obvious places like cell phones, calculators, and MP3 players. Hardly any new appliance arrives in the home without at least one controller and, most likely, there will be several—one microcontroller for the user interface (buttons and display), another to control the motor, and perhaps even an overall system manager. This applies whether the appliance in question is a washing machine, garage door opener, curling iron, or toothbrush. If the product uses a rechargeable battery, modern high density battery chemistries require intelligent chargers.

A decade ago, there were significant barriers to learning how to use microcontrollers. The cheapest programmer was about a hundred dollars and application development required both erasable windowed parts—which cost about ten times the price of the one time programmable (OTP) version—and a UV Eraser to erase the windowed part. Debugging tools were the realm of professionals alone. Now most microcontrollers use Flash-based program memory that is electrically erasable. This means the device can be reprogrammed in the circuit—no UV eraser required and no special packages needed for development. The total cost to get started today is about twenty-five dollars which buys a PICkit™ 2 Starter Kit, providing programming and debugging for many Microchip Technology Inc. MCUs. Microchip Technology has always offered a free Integrated Development Environment (IDE) including an assembler and a simulator. It has never been less expensive to get started with embedded microcontrollers than it is today.

While MPLAB® includes the assembler for free, assembly code is more cumbersome to write, in the first place, and also more difficult to maintain. Developing code using C frees the programmer from the details of multi-byte math and paging and generally improves code readability and maintainability. CCS and Hi-Tech both offer free "student" versions of the compiler to get started and even the full versions are relatively inexpensive once the savings in development time has been taken into account.

While the C language eliminates the need to learn the PIC16 assembly language and frees the user from managing all the details, it is still necessary to understand the architecture. Clocking options, peripherals sets, and pin multiplexing issues still need to be solved. Martin's book guides readers, step-by-step, on the journey from "this is a microcontroller" to "here's how to complete an application." Exercises use the fully featured PIC16F877A, covering the architecture and device configuration. This is a good starting point because other PIC16s are similar in architecture but differ in terms of IO lines, memory, or peripheral sets. An application developed on the PIC16F877A can easily be transferred to a smaller and cheaper midrange PICmicro. The book also introduces the peripherals and shows how they can simplify the firmware by letting the hardware do the work.

MPLAB®, Microchip's Integrated Development Environment, is also covered. MPLAB includes an editor and a simulator and interfaces with many compilers, including the CCS compiler used in this book. Finally, the book includes the Proteus® simulator which allows complete system simulation, saving time and money on prototype PCBs.

Dan Butler
Principal Applications Engineer
Microchip Technology Inc.

Preface

This book is the third in a series, including

- PIC Microcontrollers: An Introduction to Microelectronic Systems.
- Interfacing PIC Microcontrollers: Embedded Design by Interactive Simulation.
- Programming 8-bit PIC Microcontrollers in C: With Interactive Hardware Simulation.

It completes a set that introduces embedded application design using the Microchip PIC® range, from Microchip Technology Inc. of Arizona. This is the most popular microcontroller for education and training, which is also rapidly gaining ground in the industrial and commercial sectors. *Interfacing PIC Microcontrollers* and *Programming PIC Microcontrollers* present sample applications using the leading design and simulation software for microcontroller based circuits, Proteus VSM® from Labcenter Electronics. Demo application files can be downloaded from the author's support Web site (see later for details) and run on-screen so that the operation of each program can be studied in detail.

The purpose of this book is to

- Introduce C programming specifically for microcontrollers in easy steps.
- Demonstrate the use of the Microchip MPLAB IDE for C projects.
- Provide a beginners' guide to the CCS PCM C compiler for 16 series PICs.
- Explain how to use Proteus VSM to test C applications in simulated hardware.
- Describe applications for the Microchip PICDEM mechatronics board.
- Outline the principles of embedded system design and project development.

C is becoming the language of choice for embedded systems, as memory capacity increases in microcontrollers. Microchip supplies the 18 and 24 series chips specifically designed for C programming. However, C can be used in the less complex 16 series PIC, as long as the applications are relatively simple and therefore do not exceed the more limited memory capacity.

The PIC 16F877A microcontroller is used as the reference device in this book, as it contains a full range of peripherals and a reasonable memory capacity. It was also used in the previous work on interfacing, so there is continuity if the book series is taken as a complete course in PIC application development.

Microcontrollers are traditionally programmed in assembly language, each type having its own syntax, which translates directly into machine code. Some students, teachers, and hobbyists may wish to skip a detailed study of assembler coding and go straight to C, which is generally simpler and more powerful. It is therefore timely to produce a text that does not assume detailed knowledge of assembler and introduces C as gently as possible. Although several C programming books for microcontrollers are on the market, many are too advanced for the C beginner and distract the learner with undesirable detail in the early stages.

This text introduces embedded programming techniques using the simplest possible programs, with on-screen, fully interactive circuit simulation to demonstrate a range of basic techniques, which can then be applied to your own projects. The emphasis is on simple working programs for each topic, with hardware block diagrams to clarify system operation, full circuit schematics, simulation screenshots, and source code listings, as well as working downloads of all examples. Students in college courses and design engineers can document their projects to a high standard using these techniques. Each part concludes with a complete set of self-assessment questions and assignments designed to complete the learning package.

An additional feature of this book is the use of Proteus VSM (virtual system modeling). The schematic capture component, ISIS, allows a circuit diagram to be created using an extensive library of active components. The program is attached to the microcontroller, and the animated schematic allows the application to be comprehensively debugged before downloading to hardware. This not only saves time for the professional engineer but provides an excellent learning tool for the student or hobbyist.

Links, Resources, and Acknowledgments

Microchip Technology Inc. (www.microchip.com)

Microchip Technology Inc. is a manufacturer of PIC® microcontrollers and associated products. I gratefully acknowledge the support and assistance of Microchip Inc. in the development of this book and the use of the company trademarks and intellectual property. Special thanks are due to John Roberts of Microchip UK for his assistance and advice. The company Web site contains details of all Microchip hardware, software, and development systems. MPLAB IDE (integrated development system) must be downloaded and installed to develop new applications using the tools described in this book. The data sheet for the PIC 16F877A microcontroller should also be downloaded as a reference source.

PIC, PICmicro, MPLAB, MPASM, PICkit, dsPIC, and PICDEM are trademarks of Microchip Technology Inc.

Labcenter Electronics (www.labcenter.co.uk)

Labcenter Electronics is the developer of Proteus VSM (virtual system modeling), the most advanced cosimulation system for embedded applications. I gratefully acknowledge the assistance of the Labcenter team, especially John Jameson, in the development of this series of books. A student/evaluation version of the simulation software may be downloaded from www.proteuslite.com. A special offer for ISIS Lite, ProSPICE Lite, and the 16F877A simulator model can be found at www.proteuslite.com/register/ipmbundle.htm.

Proteus VSM, ISIS, and ARES are trademarks of Labcenter Electronics Ltd.

Custom Computer Services Inc. (www.ccsinfo.com)

Custom Computer Services Inc. specializes in compilers for PIC microcontrollers. The main range comprises PCB compiler for 12-bit PICs, PCM for 16-bit, and PCH for the 18 series chips. The support provided by James Merriman at CCS Inc. is gratefully acknowledged. The manual for the CCS compiler should be downloaded from the company Web site (Version 4 was used for this book). A 30-day trial version, which will compile code for the 16F877A, is available at the time of writing.

The Author's Web Site (www.picmicros.org.uk)

This book is supported by a dedicated Web site, www.picmicros.org.uk. All the application examples in the book may be downloaded free of charge and tested using an evaluation version of Proteus VSM. The design files are locked so that the hardware configuration cannot be changed without purchasing a suitable VSM license. Similarly, the attached program cannot be modified and recompiled without a suitable compiler license, available from the CCS Web site. Special manufacturer's offers are available via links at my site. This site is hosted by www.larrytech.com and special thanks are due to Gabe Hudson of Larrytech® Internet Services for friendly maintenance and support.

I can be contacted at the e-mail address martin@picmicros.org.uk with any queries or comments related to the PIC book series.

Finally, thanks to Julia for doing the boring domestic stuff so I can do the interesting technical stuff.

About the Author

Martin P. Bates is the author of *PIC Microcontrollers*, Second Edition. He is currently lecturing on electronics and electrical engineering at Hastings College, UK. His interests include microcontroller applications and embedded system design.

Introduction

The book is organized in five parts. Part 1 includes an overview of the PIC microcontroller internal architecture, describing the features of the 16F877A specifically. This chip is often used as representative of the 16 series MCUs because it has a full range of peripheral interfaces. All 16 series chips have a common program execution core, with variation mainly in the size of program and data memory. During programming, certain operational features are configurable: type of clock circuit, watchdog timer enable, reset mechanisms, and so on. Internal features include the file register system, which contains the control registers and RAM block, and a nonvolatile EEPROM block. The parallel ports provide the default I/O for the MCU, but most pins have more than one function. Eight analog inputs and serial interfaces (UART, SPI, and I^2C) are brought out to specific pins. The hardware features of all these are outlined, so that I/O programming can be more readily understood later on. The application development process is described, using only MPLAB IDE in this initial phase. A sample C program is edited, compiled, downloaded, and tested to demonstrate the basic process and the generated file set analyzed. The debugging features of MPLAB are also outlined: run, single step, breakpoints, watch windows, and so on. Disassembly of the object code allows the intermediate assembly language version of the C source program to be analyzed.

Part 2 introduces C programming, using the simplest possible programs. Input and output are dealt with immediately, since this is the key feature of embedded programs. Variables, conditional blocks (`IF`), looping (`WHILE,FOR`) are quickly introduced, with a complete example program. Variables and sequence control are considered in a little more detail and functions introduced. This leads on to library functions for operating timers and ports. The keypad and alphanumeric LCD are used in a simple calculator program. More data types (long integers, floating point numbers, arrays, etc.) follow as well as assembler directives and the purpose of the header file. Finally, insertion of assembler into C programs is outlined.

Part 3 focuses on programming input and output operations using the CCS C library functions. These simplify the programming process, with a small set of functions usually providing all the initialization and operating sequences required. Example programs for analog input and the use of interrupts and timers are developed and the serial port functions demonstrated in sample applications. The advantages of each type of serial bus are compared, and examples showing the connection of external serial EEPROM for data storage and a digital to analog converter output are provided. These applications can be tested in VSM, but this is not essential; use of VSM is optional throughout the book.

Part 4 focuses specifically on the PICDEM mechatronics board from Microchip. This has been selected as the main demonstration application, as it is relatively inexpensive and contains a range of features that allow the features of a typical mechatronics system to be examined: input sensors (temperature, light, and position) and output actuators (DC and stepper motor). These are tested individually then the requirements of a temperature controller outlined. Operation of the 3.5-digit seven-segment LCD is explained in detail, as this is not covered elsewhere. A simulation version of the board is provided to aid further application design and implementation.

Part 5 outlines some principles of software and hardware design and provides some further examples. A simple temperature controller provides an alternative design to that based on the mechatronics board, and a data logger design is based on another standard hardware system, which can be adapted to a range of applications—the BASE board. Again, a full-simulation version is provided for testing and further development work. This is followed by a section on operating systems, which compares three program design options: a polling loop, interrupt driven systems, and real-time operating systems. Consideration of criteria for the final selection of the MCU for a given application and some general design points follow.

Three appendices (A, B, and C) cover hardware design using ISIS schematic capture, software design using CCS C, and system testing using Proteus VSM. These topics are separated from the main body of the book as they are related more to specific products. Taken together, MPLAB, CCS C, and Proteus VSM constitute a complete learning/design package, but using them effectively requires careful study of product-specific tutorials. VSM, in particular, has comprehensive, well-designed help files; and it is therefore unnecessary to duplicate that material here. Furthermore, as with all good design tools, VSM evolves very quickly, so a detailed tutorial quickly becomes outdated.

Appendix D compares alternative compilers, and application development areas are identified that would suit each one. Appendix E provides a summary of CCS C syntax

requirements, and Appendix F contains a list of the CCS C library functions provided with the compiler, organized in functional groups for ease of reference. These are intended to provide a convenient reference source when developing CCS C programs, in addition to the full CCS compiler reference manual.

Each part of the book is designed to be as self-contained as possible, so that parts can be skipped or studied in detail, depending on the reader's previous knowledge and interests. On the other hand, the entire book should provide a coherent narrative leading to a solid grounding in C programming for embedded systems in general.

PART 1

PIC Microcontroller Systems

1.1 PIC16 Microcontrollers

- MCU features
- Program execution
- RAM file registers
- Other PIC chips

The microcontroller unit (MCU) is now big, or rather small, in electronics. It is one of the most significant developments in the continuing miniaturization of electronic hardware. Now, even trivial products, such as a musical birthday card or electronic price tag, can include an MCU. They are an important factor in the digitization of analog systems, such as sound systems or television. In addition, they provide an essential component of larger systems, such as automobiles, robots, and industrial systems. There is no escape from microcontrollers, so it is pretty useful to know how they work.

The computer or digital controller has three main elements: input and output devices, which communicate with the outside world; a processor, to make calculations and handle data operations; and memory, to store programs and data. Figure 1.1 shows these in a little more detail. Unlike the conventional microprocessor system (such as a PC), which has separate chips on a printed circuit board, the microcontroller contains all these elements in one chip. The MCU is essentially a computer on a chip; however, it still needs input and output devices, such as a keypad and display, to form a working system.

The microcontroller stores its program in ROM (read only memory). In the past, UV (ultraviolet) erasable programmable ROM (EPROM) was used for prototyping or

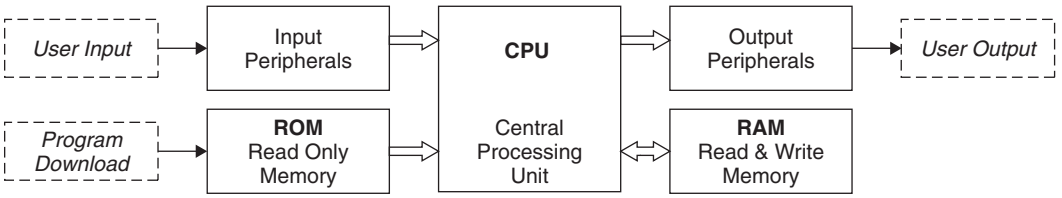

Figure 1.1: Elements of a Digital Controller

small batch production, and one-time programmable ROM for longer product runs. Programmable ROM chips are programmed in the final stages of manufacture, while EPROM could be programmed by the user.

Flash ROM is now normally used for prototyping and low-volume production. This can be programmed in circuit by the user after the circuit has been built. The prototyping cycle is faster, and software variations are easier to accommodate. We are all now familiar with flash ROM as used in USB memory sticks, digital camera memory, and so on, with Gb (10^9 byte) capacities commonplace.

The range of microcontrollers available is expanding rapidly. The first to be widely used, the Intel 8051, was developed alongside the early Intel PC processors, such as the 8086. This device dominated the field for some time; others emerged only slowly, mainly in the form of complex processors for applications such as engine management systems. These devices were relatively expensive, so they were justified only in high-value products. The potential of microcontrollers seems to have been realized only slowly.

The development of flash ROM helped open up the market, and Microchip was among the first to take advantage. The cheap and reprogrammable PIC16F84 became the most widely known, rapidly becoming the number one device for students and hobbyists. On the back of this success, the Microchip product range rapidly developed and diversified. The supporting development system, MPLAB, was distributed free, which helped the PIC to dominate the low-end market.

Flash ROM is one of the technical developments that made learning about microsystems easier and more interesting. Interactive circuit design software is another. The whole design process is now much more transparent, so that working systems are more quickly achievable by the beginner. Low-cost in-circuit debugging is another technique that helps get the final hardware up and running quickly, with only a modest expenditure on development tools.

MCU Features

The range of microcontrollers now available developed because the features of the MCU used in any particular circuit must be as closely matched as possible to the actual needs of the application. Some of the main features to consider are

- Number of inputs and outputs.
- Program memory size.
- Data RAM size.
- Nonvolatile data memory.
- Maximum clock speed.
- Range of interfaces.
- Development system support.
- Cost and availability.

The PIC16F877A is useful as a reference device because it has a minimal instruction set but a full range of peripheral features. The general approach to microcontroller application design followed here is to develop a design using a chip that has spare capacity, then later select a related device that has the set of features most closely matching the application requirements. If necessary, we can drop down to a lower range (PIC10/12 series), or if it becomes clear that more power is needed, we can move up to a higher specification chip (PIC18/24 series). This is possible as all devices have the same core architecture and compatible instructions sets.

The most significant variation among PIC chips is the instruction size, which can be 12, 14, or 16 bits. The A suffix indicates that the chip has a maximum clock speed of 20 MHz, the main upgrade from the original 16F877 device. These chips can otherwise be regarded as identical, the suffix being optional for most purposes. The 16F877A pin-out is seen in Figure 1.2 and the internal architecture in Figure 1.3. The latter is a somewhat simplified version of the definitive block diagram in the data sheet.

Program Execution

The chip has 8 k (8096 × 14 bits) of flash ROM program memory, which has to be programmed via the serial programming pins PGM, PGC, and PGD. The fixed-length

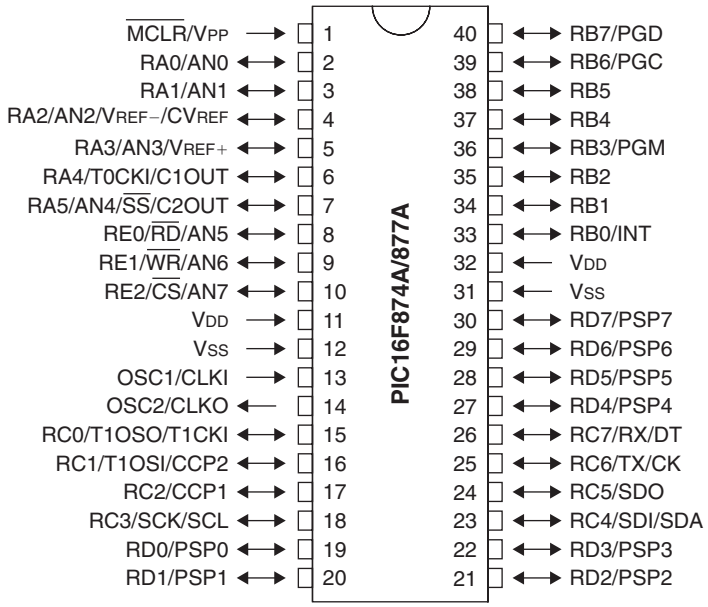

Figure 1.2: 16F877 Pin-out (reproduced by permission of Microchip Inc.)

instructions contain both the operation code and operand (immediate data, register address, or jump address). The mid-range PIC has a limited number of instructions (35) and is therefore classified as a RISC (reduced instruction set computer) processor.

Looking at the internal architecture, we can identify the blocks involved in program execution. The program memory ROM contains the machine code, in locations numbered from `0000h` to `1FFFh` (8k). The program counter holds the address of the current instruction and is incremented or modified after each step. On reset or power up, it is reset to zero and the first instruction at address `0000` is loaded into the instruction register, decoded, and executed. The program then proceeds in sequence, operating on the contents of the file registers (`000–1FFh`), executing data movement instructions to transfer data between ports and file registers or arithmetic and logic instructions to process it. The CPU has one main working register (W), through which all the data must pass.

If a branch instruction (conditional jump) is decoded, a bit test is carried out; and if the result is true, the destination address included in the instruction is loaded into the program counter to force the jump. If the result is false, the execution sequence continues unchanged. In assembly language, when CALL and RETURN are used to implement

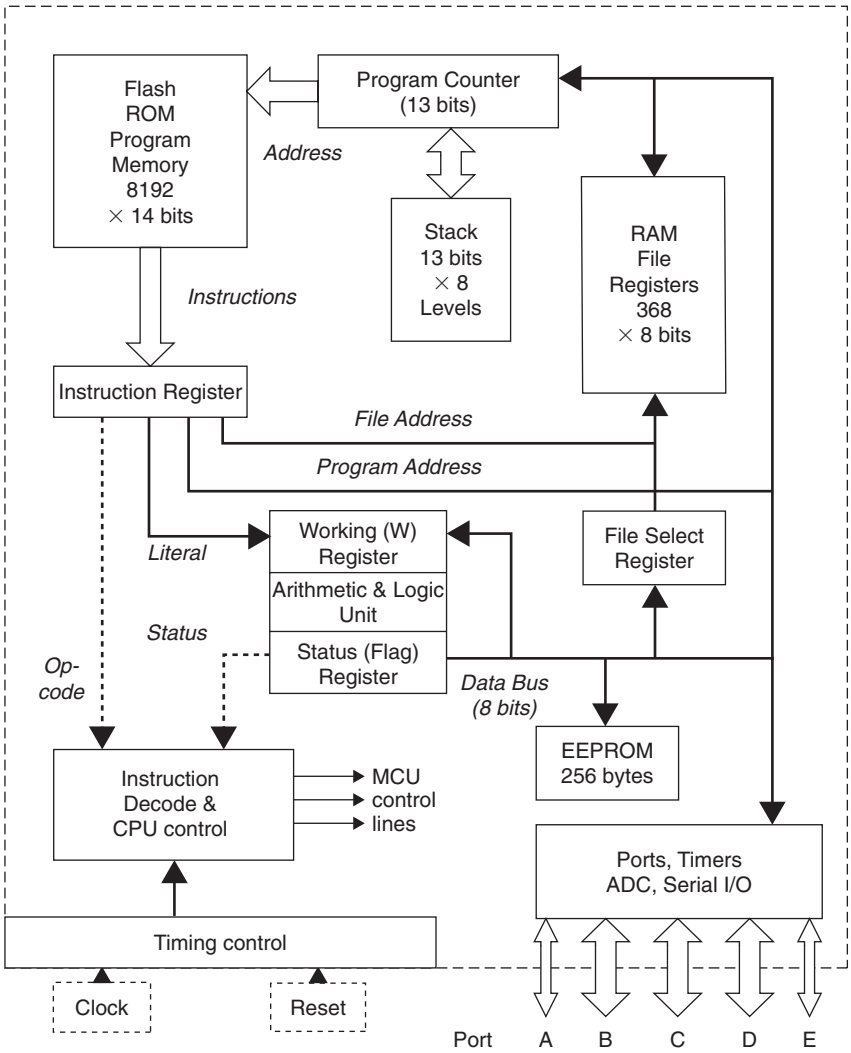

Figure 1.3: PIC16F877 MCU Block Diagram

subroutines, a similar process occurs. The stack is used to store return addresses, so that the program can return automatically to the original program position. However, this mechanism is not used by the CCS C compiler, as it limits the number of levels of subroutine (or C functions) to eight, which is the depth of the stack. Instead, a simple GOTO instruction is used for function calls and returns, with the return address computed by the compiler.