A PIC microcontroller development system for the Mac

by Francis J. Deck

Over the last decade or so, microcontrollers have become indispensible elements of electronic design. A microcontroller (MCU) is a single integrated circuit containing a CPU, RAM, input/output functions, and space for a permanently stored program. Because an MCU executes software, it can perform complex logic and timing functions, often replacing dozens of conventional logic chips or serving as the basis for "smart" electronic devices. Microcontrollers have found their way into every kind of electronic product, from consumer electronics to automotive systems.

I use microcontroller chips to design specialized data acquisition and process control hardware for both Mac and Windows systems. An MCU can simultaneously handle the data protocols of the Mac ports—especially the serial ports—and external devices such as analog-to-digital converters. A single command from the Mac can trigger a complex sequence of events in the MCU, including data reduction and closed-loop process control functions, increasing the effective throughput of the serial ports by an order of magnitude or more. In addition, an MCU is a "real time" processor that can operate continuously despite normal or catastrophic interruptions of the Mac system.

Developing for the Mac

Unfortunately, the tools required to program MCU chips are designed almost exclusively to work with IBM PC-compatible systems. Using a PC to program Mac interface hardware reminded me of the days when Mac software development required an Apple Lisa computer. To simplify interface hardware design, I designed a simple MCU development system specifically for the Mac. The system presented in this article is designed for the PIC16C6x, '7x, and '8x families of MCU chips, manufactured by Microchip Technology, Inc[1]. The PIC chips are currently the most popular MCU family of products among both hobbyists and professional developers.

My system requires both hardware and software. A very simple programmer circuit connects to one of the serial ports on a Mac. The programmer allows the Mac to write machine-language programs into the memory of a PIC chip. Readers with some experience constructing electronic circuits should be able to build the programmer in a couple of evenings. The software is a HyperCard stack that integrates a source code editor, assembler, and "front panel" for the programmer device. Readers are encouraged to download the stack, which contains supplemental documentation on the system.

Microchip gives separate specifications for "development" and "production" programming systems. The reader is cautioned that the system described in this article is for development work only, and is not suitable for commercial production. Its purpose is to permit experimentation with PIC chips from the comfort of the Mac platform. It should not be used to program microcontroller chips for applications where hardware or software failure could lead to risk of injury or damage to property.

About microcontrollers

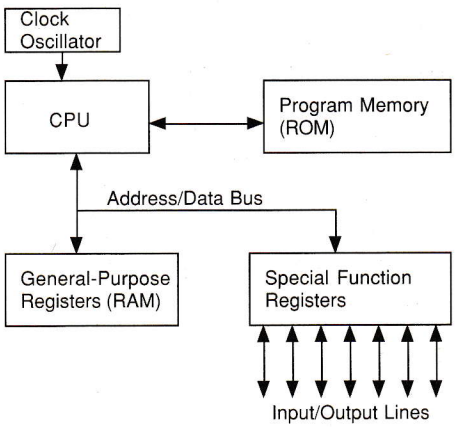
A microcontroller is a "typical" computer in that it has a CPU, memory, and input/output hardware, but MCU chips differ from the hardware of larger microprocessor systems like the Mac in some important ways. The biggest difference is one of scale: The PIC16C84 chip used to build the programmer circuit is an 8-bit processor with 36 bytes of RAM and space for 1024 program instructions. A thousand of these chips operating together could not run MacOS. Chips with more memory are available, but they are less often used due to their higher cost and increased electrical current consumption.

Memory in an MCU is organized differently than on a Mac, for two reasons: First, an MCU is expected to run exactly one program for its entire lifetime. Second, while Mac programming often involves the manipulation of large amounts of data, most MCU applications depend on tricky logic and timing algorithms that use very little RAM.

The PIC16C84 is my favorite MCU chip, because it costs around $10, and its program is stored in "electrically erasable" memory (EEPROM) that does not require an ultraviolet erasure stage between programming cycles. The PIC16C6x and '7x series use UV-erasable memory (EPROM), and come in windowed packages for development as well as opaque "one time programmable" packages that are used for commercial production.

The PIC CPU uses an instruction set that recognizes a linear array of 128 registers. Some of these registers are used as RAM, and others are virtual registers that are actually hard-wired into functions such as arithmetic flags, input/output ports, timers, and so on. The use of a register array allows members of the PIC family with different hardware features to share a single machine language. For instance, the PIC16C71 chip uses a handful of virtual registers to control an analog-to-digital converter, and the PIC16C73 adds special ser- ▶

Figure 1. A block diagram of the PIC16C84 chip shows a structure similar to a conventional microprocessor system.



Francis Deck earned his PhD in Physics at the University of Notre Dame in 1993. By day, he leads product and process development projects for a small optics company in Fort Worth, Texas. He is also actively interested in electronics, data acquisition, process control, and technical management. You may reach him via e-mail at <fdeck@aol.com>.

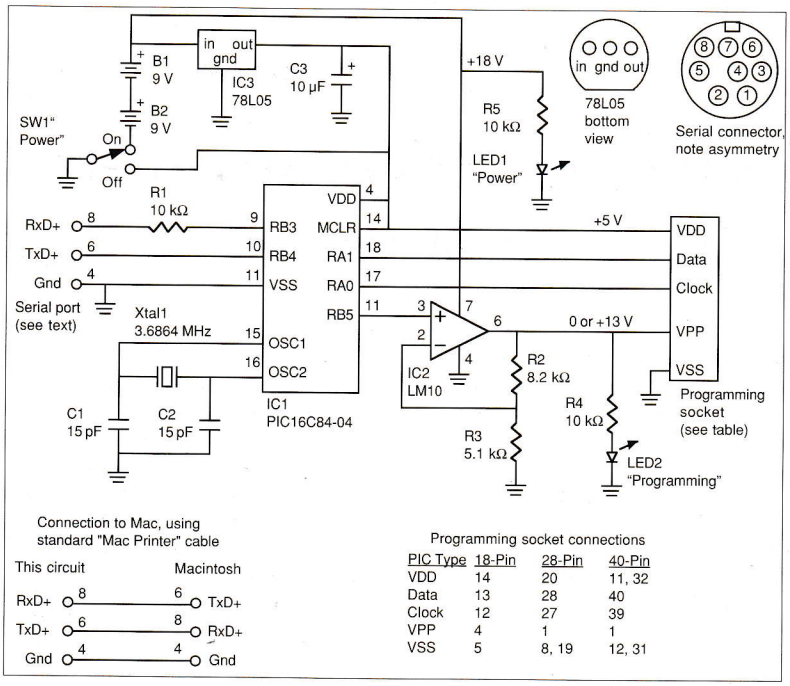


Figure 2. The PIC programmer circuit demonstrates the economy of design typical for microcontroller-based interface hardware.

ial port hardware.

Readers should be forewarned that Microchip produces some MCU families that will not be supported by the development system I describe here. At present, these include PIC16C5x and PIC17Cxx, both of which require more elaborate programming circuits.

PIC development software

I decided to adopt a single microcontroller CPU to simplify the design of both hardware and software. My system supports chips in the PIC16C6x, '7x, and '8x series, and integrates a source code editor, assembler, and "front panel" for the programmer device, into a single HyperCard stack. The stack also includes documentation that goes beyond what is provided in this article.

The powerful string-handling features of HyperCard made it easy to write a traditional two-pass assembler. In the first pass, user-defined symbols are stored in a table. These symbols are either program line labels, which are stored as addresses that could be targets of GOTO or CALL instructions, or named constants defined with the EQU (equate) directive. One other directive is supported by the assembler: ORG (origin), which overrides the default placement of instructions in consecutive memory locations. The second pass combines assembly language mnemonics with the values stored in the symbol tables to form machine language instructions.

In the development system, each programming project resides in a separate card of the stack. The project cards contain a text field for editing the program, and several buttons that select various PIC hardware options. This structure also made it easy to include several example projects with the system. Figure 1 shows a typical project card.

An XFCN resource was added to the HyperCard stack to support the Mac serial ports. One card of the stack contains buttons for operations that write, read, and verify the data on PIC chips inserted in the programming sockets. The stack can also read and write standard hexadecimal object code files for interchange of data with commercial compilers, assemblers, and programming hardware.

Programmer circuit

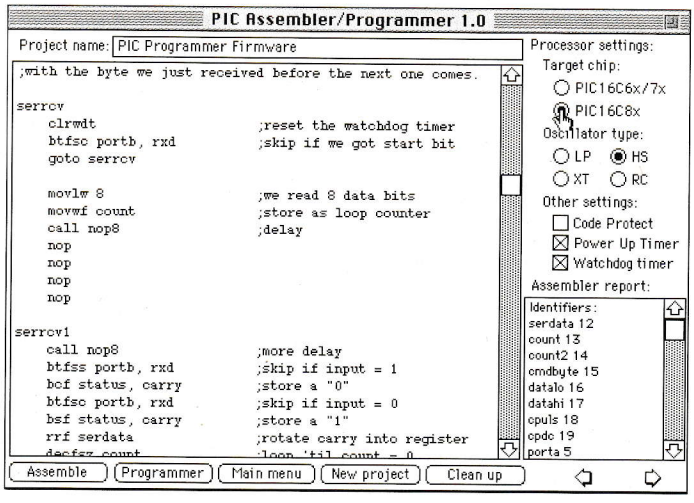
A schematic diagram for the programmer device is shown in Figure 2. This is a good example of MCU-based Mac peripheral design. A PIC16C84 chip at IC1 controls the circuit, and a clock frequency of 3.6864 MHz is required to guarantee the correct serial Baud rate. Only a single resistor, R1, is required to interface IC1 to the Mac serial port. The serial inputs on the Mac accept the 0 to +5 Volt logic levels supplied by IC1 directly. The Mac output driver generates a signal that reaches -5 V, possibly damaging IC1. A single series resistor limits currents flowing through the internal protection diodes of IC1 to levels that are well within the ratings of both IC1 and the Mac serial drivers.

The PIC16C6x, '7x, and '8x chips are programmed via three signal lines. A "data" line carries commands and data in a serial fashion, and a "clock" line supplies a series of pulses that is synchronized to the transmission of successive bits on the data line. This is referred to as a "synchronous serial" data format.

Programming data into EPROM or EEPROM requires the application of a high voltage to permit the transfer of electrical charge to locations that are physically inaccessible during normal operation of the chip. The PIC family requires a programming voltage that can be driven from 0 to +13 V. A National Semiconductor LM10 op amp at IC2 provides the gain factor of 2.6 necessary to convert the 0 to +5 V output of IC1 to the required levels. The unique high-current "rail to rail" output driver of the LM10 is essential to the operation of this circuit, so a general-purpose op amp chip can't be used in its place.

A pair of standard 9 V batteries wired in series provides power to IC2, and a standard 78L05 regulator chip at IC3 generates a +5 V supply for IC1 and for the programming sockets. It is possible for IC1 to continue to operate when the batteries are disconnected, by drawing a trickle of current through R1, so the power switch is arranged to guarantee that IC1 is properly shut down. The PIC chips come in 18-,

Figure 3. Each programming project resides in its own card within the HyperCard stack.



28-, and 40-pin packages, with correspondingly different numbers of input/output pins, and each package is connected to the programmer according to the table in Figure 1.

Readers will doubtlessly notice a "chicken and egg" situation: The programmer circuit requires a pre-programmed PIC chip! Source and object code for IC1 are provided with the software, and these are compatible with commercial PC-based tools. I imagine that many academic engineering departments have a system for programming these chips. For convenience, a printed circuit board and pre-programmed PIC chip are available from MacSciTech[2]. The circuit board has space for 18-, 28-, and 40-pin programming sockets.

The firmware has subroutines for transmitting and receiving characters via the serial port, as well as for exercising the programming sockets. A very simple command set is implemented in which each command is an alphabetic character followed by one or more binary data bytes. A detailed description of the firmware is beyond the scope of this article.

Conclusion

In operation, the HyperCard-based development package is functional but slow, due to the use of an interpretive scripting language as well as my own casual programming style. This is not a serious problem for small-scale prototyping work, since programs seldom have more than 500 lines. I struggled with the idea of converting some of the critical subroutines into XFCN resources with Heizer's CompileIt script compiler, but ultimately decided to stick with plain scripts that can be inspected by users. Although HyperCard is pretty klunky, and apparently only grudgingly supported by Apple[3], I find the speed with which it can be used to write small software applications to be unparalleled.

Compared to PC-compatible systems, the Mac suffers from a relative lack of entry-level interface and hardware development tools that is a serious weakness of an otherwise attractive platform. A microcontroller development package opens up new opportunities for creating novel Mac interface hardware designs that can be shared by users. I plan to maintain my system through the release of periodic updates that will be primarily for the purpose of fixing bugs and adding more interesting code examples.

In the January issue of the *SciTech Journal*, I will present a practical application of PIC microcontroller chips for Mac hardware interface design. With a very small handful of integrated circuits, I will implement a complete analog data acquisition system that can be used by itself or as the basis for constructing more elaborate instrumentation. I hope that my PIC development system will help to encourage Mac users to experiment with hardware design. **MST**

References

1. *The PIC16/17 Microcontroller Data Book*, published by Microchip Technology, Inc., was an invaluable resource in preparing this article. Microchip Technology, Inc., 2355 West Chandler Blvd., Chandler, AZ 85224-6199, +1 602-786-7200, Fax: +1 602-899-9210.
2. The MCU Development System is available for purchase through MacSciTech (see p.10). It includes a printed circuit board and HyperCard stack, as described in this article.
3. Our sources tell us that Apple will be releasing a new, souped-up version of HyperCard in the near future. Stay tuned to the news section of the *SciTech Journal* for more details...*ed.*

Wavelets • SPC/DOE • Microcontrollers • WWW • OS Wars

SciTech Journal™

Volume 6, Number 9
September/October 1996, $5.95 US

Official Journal of the MacSciTech Association
http://www.macscitech.org/

FEATURES

DEPARTMENTS

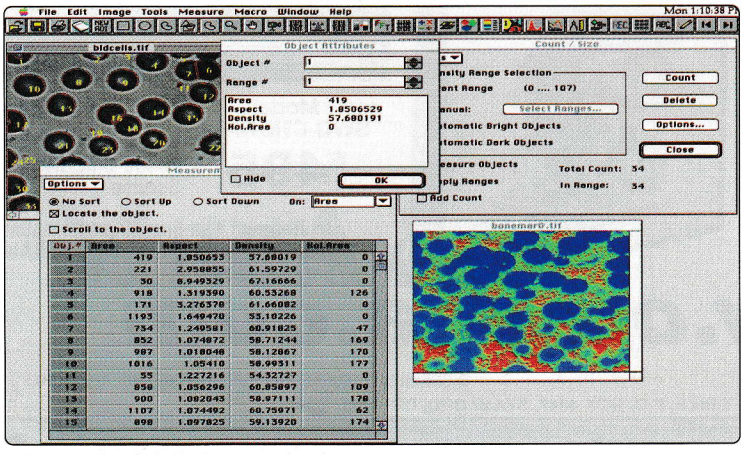


Image-Pro Plus
Image processing & analysis software
Media Cybernetics
see page 38

0 74470 90159 4 09>

49 Midgley Lane
Worcester, MA 01604-3564 USA

Bulk Rate
U.S. Postage
PAID
Fulton, MO
Permit No.38