Session 1520

EXPERIENCE OF TEACHING THE PIC MICROCONTROLLERS

Han-Way Huang, Shu-Jen Chen

Minnesota State University, Mankato, Minnesota/
DeVry University, Tinley Park, Illinois

Abstract

This paper reports our experience in teaching the Microchip 8-bit PIC microcontrollers. The 8-bit Motorola 68HC11 microcontroller has been taught extensively in our introductory microprocessor courses and used in many student design projects in the last twelve years. However, the microcontroller market place has changed considerably in the recent years. Motorola stopped new development for the 68HC11 and introduced the 8-bit 68HC908 and the 16-bit HCS12 with the hope that customers will migrate their low-end and high-end applications of the 68HC11 to these microcontrollers, respectively. On the other hand, 8-bit microcontrollers from other vendors also gain significant market share in the last few years. The Microchip 8-bit microcontrollers are among the most popular microcontrollers in use today. In addition to the SPI, USART, timer functions, and A/D converter available in the 68HC11 [6], the PIC microcontrollers from Microchip also provide peripheral functions such as CAN, I2C, and PWM. The controller-area-network (CAN) has been widely used in automotive and process control applications. The Inter-Integrated Circuit (I2C) has been widely used in interfacing peripheral chips to the microcontroller whereas the Pulse Width Modulation (PWM) function has been used extensively in motor control. After considering the change in microcontrollers and the technology evolution, we decided to teach the Microchip 8-bit microcontrollers.[1]

Several major issues need to be addressed before a new microcontroller can be taught: textbook, demo boards, and development software and hardware tools. We developed tutorials and lecture notes in which both the assembly and C languages are taught. These lecture notes and tutorials are being polished and will be published as a textbook. Three different demo boards have been designed to fit the needs of different environments. The Microchip Integrated Development Environment MPLAB® IDE is being used as the major development software. MPLAB IDE, which is free from Microchip, consists of an assembler, a linker, a simulator, and several device drivers. All three demo boards use the PIC18 microcontrollers with flash program memory and need either a programmer or a resident monitor program to download the user program onto the demo board for execution. The In-Circuit-Debugger (ICD2) from Microchip is available for performing

this task and also providing the source-level debugging capability. To make the demo board more affordable to the student, we have developed a monitor program to be programmed onto the microcontroller of the demo board. With this demo board, the user can download his/her program onto the demo board and perform in-circuit debugging of his/her program without an ICD2.

Introduction

We have been teaching the Motorola 8-bit 68HC11 microcontroller in our microprocessor courses for the last twelve years. Motorola stopped all new development of the 68HC11 family for about ten years. The features developed after the late 1980s, including the PWM and interfaces such as CAN, I2C, USB, and LIN, were not included in the 68HC11 family microcontrollers and will not be included in any 68HC11 members because Motorola has shifted its development resources to other microcontroller families such as the 68HC908 and the HCS12.

The interconnection capability is important in today's embedded products. The 68HC11 is quite limited in this capability because it provides only the USART and SPI functions. The I2C and USB interfaces are for general-purpose use, whereas the CAN and LIN buses have become popular in automotive, factory automation and process control applications. We felt the need to introduce these subjects to students, but the subjects cannot be experimented by using the 68HC11 microcontroller.

Although the choice of the microcontroller taught in a class should not be solely determined by the market dominance, we did feel that learning a microcontroller that is popular in the marketplace will help students' employment opportunities. When the market leader also provides most of the newer features that we like to cover in the course, its adoption offers many benefits. Thus, we decided to teach the Microchip 8-bit microcontrollers to our students.

The Microchip PIC Microcontrollers

Microchip produces hundreds of different microcontrollers, which are further divided into several families of architectures. The PIC16 and PIC18 are the two major families. The architectures of these two families are similar. Many of their peripheral modules have similar register layouts and functions. However, the PIC18 was designed to remove the limitations embedded in the PIC16 family, improve performance, and simplify the programming of all peripheral functions. [1]

The PIC16 Family

The PIC16 devices have been well-established in the market and are widely used in the industry and by the hobbyist. New devices of this family continued to be introduced with more memory and newer peripheral features. Among them [1] are

- General I/O
- Parallel slave port

- Timers with capture, compare, and pulse-width modulation modules
- USART, SPI, and I2C modules
- 10-bit A/D converter

Some of the new PIC16 devices are designed to support special interfaces such as the LIN and USB buses.

The PIC16 CPU is pipelined and provides 35 instructions. The PIC16 architecture divides the program memory into pages and divides the data memory into banks. The registers of peripheral functions spread into several memory banks. The users are required to keep track of the program memory page and the data memory bank that are in use. Therefore, the assembly language programming has been a frustrating experience for our students. To reduce the frustration of programming for students, we introduced C language programming for the PIC16 family after a brief introduction of assembly programming. C language programming allows students to work on the program logic at a higher level and at the same time be able to manipulate the peripheral registers found in assembly language. The PIC16 family supports only 8KB of program memory space and hence is only suitable for small and simple applications. The freeware PICC-Lite C compiler from Hi-Tech Software was used to compile all the C programs written for the PIC16 family.

Because of the architecture of the device, it has been difficult to create a debug monitor for the PIC16 microcontroller. The PIC16 background debugger architecture prevents the use of subroutine calls and returns. Execution of the return instruction takes the processor out of debug mode and back to user mode. The PIC16 has an eight-entry return address stack, which proves to be difficult if not impossible to be shared between the user program and the monitor program.

The PIC18 Family

The design of the PIC18 microcontroller [2,3,4] has eliminated the need for selecting the active program memory page. The PIC18 program counter and the "call" and "goto" instructions contain as many address bits as are needed to address the complete program memory space. The program is able to jump to any place within the 2-MB memory space, making the program memory paging in the PIC16 family unnecessary.[1]

Although the data memory is still divided into banks, the PIC18 microcontroller provides three File Select Registers (FSRs) and the "move from file register to file register (movff)" instruction to provide access to the whole data memory space. All the indirect data memory accesses can be performed without specifying the bank. A special **access bank** is created, which contains the first 96 bytes of the SRAM (128 bytes for a smaller number of PIC18 devices) and the last 160 bytes of special function registers (128 bytes for a smaller number of the PIC18 devices). The access bank has eliminated the need for data bank switching completely for the special function registers (excluding devices with the CAN module) and for programs using less than 96 bytes (or 128 bytes for some devices) of direct access RAM.

The PIC18 microcontroller provides a 31-entry return address stack but does not support a user data stack structure directly in the hardware. However, one can implement the data stack using one of the three FSR registers as the stack pointer in software. Data stack is very useful in parameter passing and local variable allocation during the subroutine call. One can also use another FSR register as the frame pointer to facilitate the access to the data in stack. The combination of incoming parameters, saved registers, and local variables is referred to as the **stack frame** [5, 6]. The PIC18 stack frame is shown in Figure 1.

The PIC18 family has many improvements in its implementation of peripheral functions over the PIC16 family. The timer function is one good example. The 8-bit architecture prevents the PIC18 microcontroller to read a 16-bit value in one operation. However, the PIC18 microcontroller will copy the upper byte of a 16-bit timer into a latch when its lower byte is being read. This makes sure that the 16-bit value that the user reads belong together. Some of the PIC18 members improved their PWM function by allowing the user to program a deadband into the PWM output which can then be used to drive an H-bridge easily. The H-bridge is commonly used in motor control circuit. The CAN bus module is also added to a group of PIC18 members.

Microchip provides a PIC18 C compiler that is reasonably priced for university use. Several third party software companies also sell C compilers and IDEs for the PIC18 microcontrollers.

After teaching both the PIC16 and the PIC18 families of microcontrollers, we decided to focus on the PIC18 family for the obvious advantages in doing so.

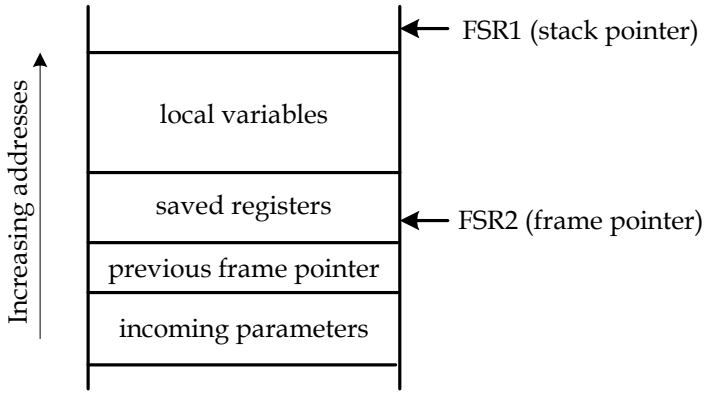

Figure 1. PIC18 Stack frame

1

The Textbook

Comprehensive and well-written textbooks were not available when we started to teach the PIC microcontrollers. We developed a set of lecture notes and tutorials for teaching the PIC microcontrollers. These tutorials cover from the introduction, architecture, and general programming to the application circuits and programming of all peripheral functions. Both the assembly and the C languages were used throughout the tutorials and lectures. A subset of the tutorials that focus on the PIC18 microcontroller are being polished and revised and will be published as a textbook. We believe, as more and more schools start to teach the PIC18 microcontrollers that other authors will also become interested in writing textbooks for the PIC18 microcontrollers [5,7].

The Development Boards

Many PIC microcontrollers have on-chip flash memory. It is easy to build a simple development board with a few additional components. Students could construct the circuit on a solderless proto-board in about an hour. We have been successfully running the student lab with these self-constructed development boards for several semesters.

One problem with this approach is the reliability of the self-constructed development board. The quality problem tended to haunt the students for the rest of the semester. A pre-fabricated development board with some built-in I/O interfaces allows students to get started and gain some confidence in a short period of time.

Three versions of the development board were developed for different teaching environments. The first development board contains a 28-pin ZIF socket and a 40-pin ZIF socket, which allows the user to experiment with any 28-pin or 40-pin PIC18 microcontroller (for example, PIC18F452 and PIC18F252). The second development board uses the PIC18F8680 as its controller and has a CAN bus interface. The third development board uses the PIC18F8720 as its controller. This development board is designed for those who need more program memory (128KB is available) to hold their applications. All three demo boards provide the following features:[1]

- Digital signal outputs with frequencies ranging from 1 Hz to 8 MHz
- Temperature sensor with SPI interface
- EEPROM with I2C interface
- Time-of-day chip with SPI interface
- Eight LEDs
- A 2 x 20 LCD display
- Two or four debounced key switches
- 4 x 4 keypad connector
- One potentiometer
- A rotary encoder
- Speaker connector
- DIP switches for input

With these development boards, students could get familiar with the microcontroller programming and interfacing before moving on to design their application projects. Information about these development boards is available from the website at www.evb.com.tw.

The Software Development Tools

The MPLAB IDE from Microchip is our main software development tool. The MPLAB IDE is a software development environment that consists of

- a context-sensitive text editor
- an assembler
- a linker
- a simulator
- device drivers for in-circuit emulators, in-circuit debuggers, and programmers
- interface to C compilers from Microchip and other vendors

These tools together allow students to develop software and perform debugging using the simulator all through the same user interface. The MPLAB IDE generated the output in the Intel Hex format, which the user can program it into the target device.

We mainly used the PIC devices with on-chip flash memory for their quick erasure and reprogramming capability. Device programmers are required to perform the programming task. We quickly found that many programmers are needed to support students' programming activity. A bootloader program was written to download the program to the microcontroller in order to reduce the demand for the programmer hardware. This bootloader evolved into the debug monitor program described in the next section.
[1]

C compiler

Microchip Technology does not provide a C compiler for the PIC16 family microcontrollers. We used the free demo version of C compiler from Hi-Tech Software for the PIC16 family with reasonable success. Most of the constrains that we encountered were caused by the PIC16 architecture and memory size limitations rather than by the compiler itself. If larger and more flexible memory usage is required for the project, then the PIC18 family devices should be seriously considered.

We use the Microchip C compiler for the PIC18 family microcontrollers. The sample C programs in the Huang's PIC18 textbook [5] were written for and tested with the Microchip C compiler.

Programming in flowchart

One of the obstacles we faced in teaching programming to the electronics students is that they often are bogged down by the complexity and rigidity of the syntax of the language

and overlook the essence of programming, the algorithm. Especially for our ET students, they do not have much of the programming experience before the microprocessor course. We found programming in flowchart a good way of introducing students to algorithms and the Flowcode a good tool to allow students immediate hardware experiences with their programs.

Flowcode is a product of Matrix Multimedia [9]. It offers a GUI for programmer to draw flowcharts. The Flowcode compiler compiles the flowchart into C code, assembly code, and binary files for device programming. The Flowcode IDE has a single command to execute the flowchart compiler, the C compiler, the assembler, and the device programmer. Given proper hardware, the device is programmed and executed on the target circuit.

Flowcode also has embedded C program or assembly program snippets in a process box of the flowchart. This mechanism allows us to gradually expose students to programming languages without overwhelming burden of understanding the whole language construct. We are currently preparing for an ET introductory microcontroller course using Flowcode. If successful, we will extend the Flowcode use to our EET program.

Flowcode currently supports most of the PIC16 family devices. Although Matrix Multimedia has no immediate plan to extend it to cover PIC18 family, programming in flowchart and C code is largely device independent.

The Debug Monitor Program

Although the MPLAB SIM simulator is convenient for debugging the software, eventually the program has to be tested on the target circuit. Microchip offers in-circuit emulator (ICE) and in-circuit debugger (ICD). The in-circuit emulator is expensive and requires a dedicated pod for each different type of microcontroller. It has value in introducing students to the use of a microcontroller emulator. However, the cost of an emulator is not justified for general lab use.

The less expensive ICD uses the background debug mode (BDM) to implement the instruction breakpoint, instruction stepping, and memory display and modification. The ICD communicates with the target microcontroller via a serial interface, which can be slow sometimes.

It is very desirable for students to own their demo boards so that they can experiment the hardware and software at their residency. However, the cost of the combination of a development board and an ICD is prohibitive for most students. Therefore, we developed a debug monitor program to be programmed onto the microcontroller of the demo board. The monitor communicates with a terminal program running on a PC and provides the following functions [8]:
[1]

- Display the contents of the on-chip SRAM, EEPROM, and program memory
- Modify the contents of the on-chip SRAM, EEPROM, and program memory
- Disassemble the program in the program memory
- Allow the user to enter instructions onto the program memory
- Download the user program (hex file) onto the demo board from the PC
- Set breakpoint
- Single step instructions
- Dump stack trace

This monitor program provides students a development environment without the need of an ICD, which would significantly reduce the cost of owning a PIC18 development suite. With this monitor program, students can purchase only a demo board and work on the projects in their residency.

The monitor program uses 8K bytes of program memory and one bank (256 bytes) of data memory. That leaves 24K bytes of program memory and 1K bytes of data memory for users with the PIC18F452, the smallest memory device of the development boards we have. The monitor program uses only one dedicated pin (for external program halt) and two SCI pins that could be shared with the user programs.

This monitor has a limitation: it allows the user to set one breakpoint at a time. We are studying the possibility of allowing the user to set multiple breakpoints and also consider how to add a graphical user interface that can provide symbolic debugging capability. [1]

Conclusion

We determined to teach Microchip 8-bit PIC microcontrollers for three main reasons: (1) Motorola has stopped new development for the 68HC11 microcontroller that we taught, (2) the PIC16 and PIC18 microcontroller provide all the peripheral functions that we like to teach, and (3) Microchip is one of the leaders in the 8-bit microcontroller market. Preparing for teaching a new microcontroller is nontrivial. It includes the preparation of teaching materials and acquiring software and hardware development tools. We prepare lecture notes and tutorials for both the Microchip PIC16 and the PIC18 microcontrollers. The teaching materials for the PIC18 microcontroller are being polished and will be published as a textbook. Both the assembly and the C languages were used throughout the book. We wish students gain more insight through learning the assembly language programming and be productive by programming in the C language.

We designed three PIC18 development boards to fit the needs of different environments. One of the development boards has both the 28-pin and 40-pin ZIF sockets and hence allows the user to experiment with all of the 28-pin and 40-pin PIC18 microcontrollers. The second development board has a CAN bus interface and allows the user to experiment with the CAN bus. The third development board has 128KB of flash memory and can handle larger applications. All three development boards have similar peripheral chips.

Unlike the Motorola 68HC11, the Microchip PIC16 and PIC18 do not have debug monitors for the general users. The user would need to use either a dedicated programmer or the ICD to download the program onto the development board for execution. This additional cost prevents students from being able to purchase their development suites to enhance the learning of the PIC microcontroller. We develop a debug monitor to solve this problem. With this monitor, students will be able to download their programs onto the demo board for execution without using a dedicated programmer or in-circuit debugger. This monitor allows the user to display and modify memory contents, set a breakpoint, and step through the program. These functions can facilitate the program debugging process.

References

1. "PIC16F87XA Data Sheet", Microchip, Chandler, AZ, 2001
2. "PIC18FXX2 Data Sheet", Microchip, Chandler, AZ, 2002
3. "PIC18FXX20 Data Sheet", Microchip, Chandler, AZ, 2002
4. "PIC18F6585/8585/6680/8680 Data Sheet", Microchip, Chandler, AZ, 2003
5. "The PIC18 Microcontroller—An Introduction", Han-Way Huang, Delmar Thompson, Clifton Park, New York, 2004
6. "MC68HC11: An Introduction", Han-Way Huang, Delmar Thompson, Clifton Park, New York, 2002
7. "Embedded Design with the PIC18F452 Microcontroller", J. Peatman, Prentice Hall, Upper Saddle River, New Jersey, 2003
8. "User's Manual for the PIC18MON Debug Monitor", Shujen Chen, 2004
9. "Flowcode datasheet", Matrix Multimedia Limited, http://www.matrixmultimedia.co.uk/

1

HAN-WAY HUANG, PH. D., is a Professor in the Department of Electrical and Computer Engineering and Technology at Minnesota State University in Mankato, MN. Before that, he worked in the computer industry for four years. He received his M.S and Ph. D. degrees in Computer Engineering from Iowa State University. He has taught microprocessor and microcontroller for more than 16 years and authored several books on microprocessor applications.

SHU-JEN CHEN is an Assistant Professor of Electronic Engineering Technology at DeVry University, Tinley Park, IL for three years. Prior to that, he worked for Bell Labs as a research and development engineer for sixteen years and University of Wisconsin-Madison as an academic staff for three years. He received his masters in electrical engineering from University of Wisconsin-Madison. His main interest is in microprocessor and microcontroller applications.