Designing a Embedded Controllers Course Utilizing a Low Cost 8051 Micro-Controller Board

Kenneth E. Dudeck, Associate Professor of Electrical Engineering

Pennsylvania State University, Hazleton Campus

Abstract

This paper describes an innovative Microprocessor/Microcontrollers course that is currently being taught at the Hazleton Campus of Penn State University. The course utilizes a commercially available 8051 Microcontroller Development Board. This 8051 development board provides an easy and low-cost way to develop projects based on the 8051 microcontroller without the need to purchase any other equipment, such as EPROM programmers or emulators. This is accomplished by a resident MONITOR boot-up program that acts like the development board's operating system. This MONITOR program is preprogrammed in the 8051 FLASH ROM.

This paper will outline some examples of using this board as well as illustrate an off line 8051 simulator that students can use to develop and test their programs outside of class.

Finally this paper will show how to use the board as a dedicated embedded controller by modifying the boot up sequence so that the MONITOR is by-passed and the board begins execution of the user program, stored in flash ROM, upon power up.

Introduction

Microprocessors are taught in a three credit course with an embedded laboratory hour in the two-year associate Electrical Engineering Technology (2EET) program at the Penn State University.

Prior to taking this course, students are required to take a three credit digital electronics course with an accompanying lab that provides the necessary background needed to understand microprocessors.

Although a large majority of the course is devoted to Assembly Language Programming, there is a substantial emphasis on the hardware needed to design a minimal micro-computer system. If the course were strictly intended for Computer Science majors, the choice of the Intel 8086/8088 processor would be appropriate for PC based programs. But for EET applications, a good microcontroller is the logical choice. In this course, the Intel 8051 is studied.

The equipment and software needed to teach this course is summarized below.

1) An 8051 Microcontroller Development Board

"Proceedings of the 2003 American Society for Engineering Education Annual Conference &Exposition

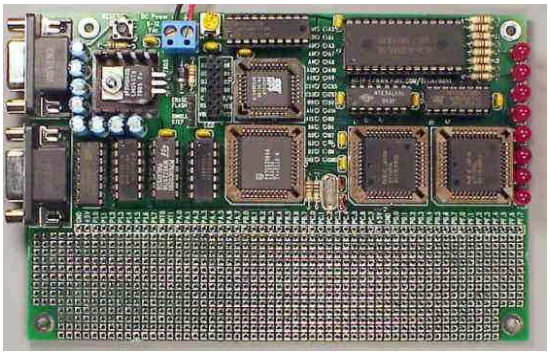
2) An 8051 Assembler
3) An 8051 Simulator

The board used here is available for purchase on-line and costs under $80, while the assembler and simulator are freeware and available on-line as well. The assembler used works well with both the board and the simulator and provides the students with the opportunity to develop and test their programs outside of lab. Some students have elected to purchase the board for themselves and incorporate it as a dedicated controller in required projects in other EET courses.

All the necessary software, documentation, and written labs needed to offer the course, as described herein, are available from the author's webpage. [1]

Microcontroller Board Details

The board used in the course is an 8051 Microcontroller Board developed by Paul Stoffregen, shown below. This board is designed around the Amtel AT89C52 chip. The AT89C52 is actually an 8052 processor with the on-board 8K flash ROM preprogrammed with a monitor utility program called "PAULMON2 and 82C55 (External 4 Port Expansion Chip).



The chip also provides four internal I/O ports, 256 internal RAM locations, and two serial ports. Other devices on the board include a 6264 (8K External RAM Chip), a SST39F512 (8K External Flash ROM), and 82C55 (External 4 Port Expansion Chip). Other devices on the board include a 6264 (8K External RAM Chip), SST39F512 (8K External Flash ROM), and a 82C55 (External 4 Port Expansion Chip).

The two 8052 internal serial ports are designed and configured as RS232 serial communications ports that provide for connections to a PC using a terminal emulation such as window's "Hyper- terminal". In this way, programs are written using a text editor, then assembled, and downloaded into RAM or FLASH ROM using the MONITOR utility. For more information on the board and software details, visit the PJRC homepage[2].

The memory map for the board is shown below.

8051 Microcontroller Board Memory Map

Address Space	Device	
0000H – 1FFFH	8K ROM with PAULMON2 permanently stored.	Amtel AT89C52 Chip
00H – FFH	Internal RAM	

Port 0 –Port 3 Internal I/O Ports (port 1 direct)

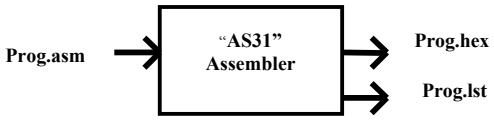
2000H – 3FFFH 8K External RAM (user programs) 6264 Chip
4000H – 4003H Port Expansion 82C55 Chip
4000H -Port A
4001H – Port B
4002H – Port C (on board LED's)
4003H –Port Program Byte

8000H – 9FFFH 8K External Flash ROM SST39F512

The board also provides experimental workspace for custom use and provides all the data, address, and control bus signals for custom expansion. In this course, Port A was interfaced to 8 SPST DIP switches. The board comes with 8 LED indicator lights pre-wired to PORT C. This set up provides for simple Input/Output experimentation.

Assembler Details

The 8051 assembler used in the course is called "AS31.exe". It is a free 8051 DOS based assembler originally written by Ken Stauffer. Both the simulator and the board require 8051 assembled programs to be downloaded using the standard Intel Hex file format. The advantage of this assembler is that it takes assembly language text files, stored with an ".ASM" file extension (i.e. "prog.asm") and produces the corresponding machine code in the ".HEX" file format. This process begins with writing the 8051 assembly program using a text editor, such as window's "Notepad" and then saving it as an *.ASM file. The assembler then produces the .HEX file and a list file as shown below.



To evoke the assembler, the user must use the DOS command prompt and make sure that the working directory contains the "as31.exe" application file as well as the "prog.asm" file. The user then types:

C:> *as31-l prog.asm*

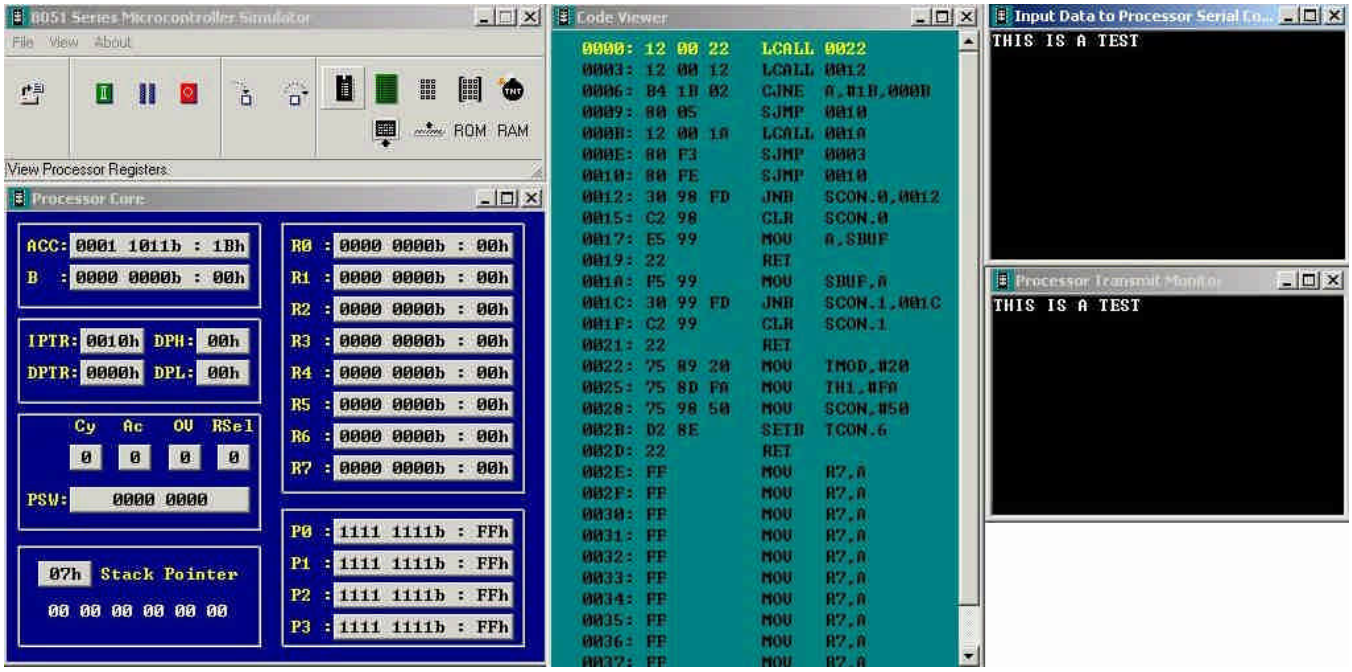
The –l option produces the LST file which contains both the assembly and machine language files. If there are no errors in the assembly file, the corresponding hex file is generated and ready to be downloaded to either the board or the simulator.

Simulator Details

The 8051 freeware simulator used is written by Steve Nolan and is a nice way to develop 8051 programs off line. The simulator will only read programs that have been assembled and converted into an Intel Hex format file by using the as31 assembler

Once the hex files have been generated, they can be loaded into the simulator and run.

The simulator interface is shown below.



Notice that the simulator interface shows all the 8051 registers, flags, ports and provides simulated keyboard and terminal interfaces via the SBUF register. For example the image above shows the simulator loaded and running a program the reads the simulator keyboard buffer and sends the ASCII value to the simulator display until the "ESC" key is pressed. With minor modifications this program can be loaded onto the Micro-controller board and run in lab.

Using the 8051 Microcontroller Board

The board is connected to the PC via the serial port COM1. The windows application "Hyper-terminal" is run and a new connection is created. Once the board is powered up, hitting the ENTER key while in Hyper-termial, causes the PAULMON2 utility to run and begin accepting user commands. An assembled hex file is downloaded to the board using Hyper-terminal's "Send Text File" feature. Once downloaded into the board's memory, the program can be run using the PAULMON2's "R" command.

When writing the assembly program that will be sent to the board, a special 64 byte header file must be included at the top of .ASM file. This header file is needed for the

PAULMON2 utility to properly manage the running of the program. This header file serves the same purpose as the header file in an ".EXE" file that is run under the window's operating system.
The header file is shown below:

```
.equ    locat, 0x2000          ;Location for this program
.org    locat
.db     0xA5,0xE5,0xE0,0xA5    ;signiture bytes
.db     35,255,0,0             ;id (35=prog)
.db     0,0,0,0                ;prompt code vector
.db     0,0,0,0                ;reserved
.db     0,0,0,0                ;reserved
.db     0,0,0,0                ;reserved
.db     0,0,0,0                ;user defined
.db     255,255,255,255        ;length and checksum (255=unused)
.db     "YOUR PROGRAM NAME HERE",0      ;max 31 characters, plus the zero
.org    locat+64               ;executable code begins here
```

There are three elements of the header file that can be adjusted to direct PAULMON2 on how to manage the particular program.

Line 1: .equ locat, 0x2000 - This sets the memory location of where the program will be downloaded into memory. As shown, this program will be downloaded beginning at the first RAM location in the memory map. If the program is to be downloaded into FLASH ROM, the line would be set to:

.equ locat, 0x**8000**

Line 4: .db 35,255,0,0 - This determines if the program will run under the PAULMON2 operating system or if the program will self boot upon power supplied to the board. As shown, the program will be run under PAULMON2 and Hyper-terminal prompting.

In order to make the board run the program immediately upon power up of the board, as in a dedicated embedded controller application, the line must be changed to:

.db **249**,255,0,0

Once this self-booting program has been downloaded into memory, the PAULMON2 program will never again to be able to be run, since it is always [1]bypassed upon board power up.

The flash ROM must be manually erased by shorting the "ERASE FLASH" jumper and pressing the board's reset button. The board will now boot-up to PAULMON2 as before.

Line 11: .db "*YOUR PROGRAM NAME HERE*",0 - This line is used as a program label for PAULMON2. Whatever is typed between the quotes will be listed as the program name when evoking the 'R' command.

Simple Discrete Input and Output on the Board

The following program illustrates simple input and output using the external ports on the 82C55 chip. The program will read the 8 DIP switches wired at Port A and send that value to the LED's connected at Port C. The LED's are turned on by corresponding 0's so the accumulator must be complemented before the value is sent to Port C. The program will not allow for all eight LED's to be on at the same time.

```
.equ    locat, 0x2000          ;Location for this program
.equ    port_a, 0x4000         ;82C55 memory locations
.equ    port_b, 0x4001
.equ    port_c, 0x4002
.equ    port_abc_pgm, 0x4003
.org    locat
                [Header file goes here]
          mov       dptr, #port_abc_pgm ; configure the 82C5
          mov       a, #90H
          movx      @dptr, a ;Port A in, B and C out
loop:     mov       dptr, #port_a          ; read switches at Port A
          clr       a
          movc      a, @a+dptr
          cjne      a,#0FFH,cont       ; quit before turning on the last LED
          sjmp      exit
cont:     mov       dptr, #port_c      ; send value to Port B
          cpl       a
          movx      @dptr,a
          sjmp      loop
exit:     ret ; return to PAULMON
```

When this program is downloaded and run on the board, as a switch is closed the corresponding LED will go on. This program will run in an endless loop until all the switches are closed. The moment the last switch is closed, the program will terminate back to the PAULMON2 program, leaving only seven of the LED's on.

Conclusions

This paper has shown an inexpensive way to illustrate the concept of microprocessors. By using a microcontroller board that has its own development system built in, no additional ROM burners are needed to program the board's memory. A single board is all that is needed program and run the student's application. By including a freeware 8051 simulator, students can gain the necessary programming experience outside of class. The assembler used herein works well with both the board and the simulator.

Finally, this board can be configured to be used as an embedded controller. This offers the student an affordable way of incorporating this board into any existing student project where a dedicated controller may be required.

Bibliographic Information

[1] http://www.hn.psu.edu/faculty/kdudeck/EET211/ET211SY.htm

[2] http://www.pjrc.com/tech/8051/

Biographical Information

KENNETH DUDECK is an Associate Professor of Electrical Engineering at the Pennsylvania State University located in Hazleton PA 18202. He has been teaching Electrical, Computer, and Electrical Technology Engineering Courses for the past 15 years. He is also a consultant for the Naval Air Warfare center in Patuxent River, MD.