UVI51: A SIMULATION TOOL FOR TEACHING/LEARNING THE 8051 MICROCONTROLLER

Alfredo del Río[1] and Juan José Rodríguez Andina[2]

Abstract *¾ Teaching/learning microcontrollers in the laboratory has been traditionally carried out using general purpose simulators and/or evaluation boards. In-circuit emulators are not widely used because their high cost. This paper presents UVI51, a software tool developed for teaching/learning the 8051 microcontroller in the laboratory and/or the classroom. UVI51 includes an assembler, a multimicro simulator, a logic analyzer, and an assistant. The tool allows to simulate systems consisting of up to 4 microcontrollers plus a set of external peripherals. Both the CPU core and the embedded peripherals of each microcontroller are simulated. Everything in UVI51 has been designed with the educational perspective in mind. A set of windows depict the configuration and behaviour of every embedded peripheral. UVI51 is currently being used in several courses on microcontrolllers at University of Vigo (Spain) and also at the college level. The tool is suitable for learning nearly everything about the 8051, ranging from the CPU and instruction set basics to complex use of timers, interrupts and the serial port. This paper shows the benefits of using UVI51 as an alternative to traditional instruction tools.*

Index Terms ¾ Education, MCS51, microcontrollers, simulation.

INTRODUCTION

8-bit microcontrollers are widely used for introductory courses on this topic mainly because they are simpler and easier to describe than 16-bit or 32-bit microcontrollers, and also due to their low cost. Among them, Motorola 68HC11, Microchip PIC family, and the industry standard 8051 are prevalent [1].

Even though many simulators are available for the 8051 family [2][3][4], most of them do not support multimicro operation, nor simulate all the embedded peripherals, and only a few can simulate external peripherals. In our opinion none of them is a CAI-oriented simulator, since they do not include features as 8051-specific graphical help windows. Furthermore, the most complete one [4] is too expensive for many educational environments. These limitations recommended the development of a new simulator within the universitary scene, providing a free tool to the education community.

This paper presents a software tool, called UVI51, specifically designed for teaching and learning the 8051 microcontroller. The current version (5.0) is the result of succesive enhancements carried out on early versions [5]. It allows to simulate systems consisting of up to 4 microcontrollers operating concurrently, plus a set of interconnected external peripherals.

UVI51 consists of an assembler, a multimicro simulator (the core of the tool), a logic analyzer, and an assistant that guides the user through the development process. UVI51 provides its own assembler, but standard assemblers as [6] can also be used. The C language is also supported, but in this case a standard compiler, as [7], must be used.

The structure of the paper derives from our aim of highlighting the educational worth of UVI51. Therefore, the main components of the tool (assistant, assembler, simulator and logic analyzer) are described in the following sections. Furthermore, a section is devoted to an application example. Finally, the conclusions of the work are presented.

THE ASSISTANT

The assistant guides the user through the development process which basically consists of the following steps: i) System definition; ii) Source code writing; iii) Code assembly; iv) Simulation.

The system to be simulated is first specified by editing a system configuration file. This is an ASCII file where the user defines every microcontroller indicating their respective source file names, clock frequencies, and memory sizes. Furthermore, the user can include external peripherals from a predefined set. Finally, the user indicates the interconnections between the microcontrollers and the external peripherals. An example of a system and its corresponding configuration file is shown in Figure 1.

The present version of UVI51 supports the following types of external peripherals: i) Pushbuttons; ii) Switches (either manually-operated or node-controlled); iii) Light-Emitting Diodes (LEDs); iv) 7 segment displays; v) LCD screens (either serial or standard types as Trident MDLS 16265). Complex peripherals can usually be simulated using an additional microcontroller with a suitable program.

Both assembler and C language are supported as source code, but a standard compiler is needed for the latter.

[1] Alfredo del Río, University of Vigo, Dpto. Tecnología Electrónica, Vigo, Spain, ario@uvigo.es
[2] Juan José Rodríguez Andina, University of Vigo, Dpto. Tecnología Electrónica, Spain, juanjo@dte.uvigo.es

0-7803-6424-4/00/$10.00 © 2000 IEEE **October 18 - 21, 2000 Kansas City, MO**
30th ASEE/IEEE Frontiers in Education Conference

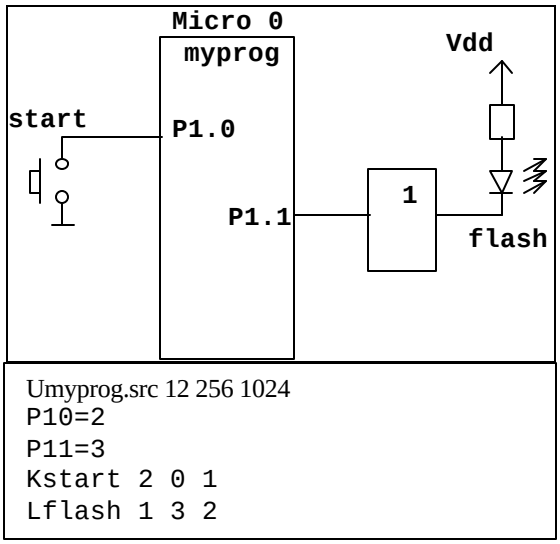

FIGURE. 1
SYSTEM BLOCK DIAGRAM AND ITS CORRESPONDING CONFIGURATION FILE.

THE ASSEMBLER

UVI51 includes an assembler that eliminates the need for a third-party assembler. The assembler generates executable code directly, in Intel-Hex format; therefore no relocatable expressions are accepted. It does not support macroinstructions. If some of these features are needed, a commercial asembler compatible with Intel´s asm51 can be used.

UVI51 assembler makes use of the same syntax as Intel's asm51, with a few exceptions. In addition, conditional assembly is supported. Computable expressions can be used wherever a numeric value is allowed.

THE SIMULATOR

The simulator is the core of UVI51. Its main features are: i) Multimicro capability; ii) Embedded and external peripherals simulation; iii) Trace into, step over and continuous simulation; iv) Unlimited breakpoints; v) Additional breakpoint by value; vi) Chronograms generation; vii) Graphic user interface; viii) Help windows for every embedded peripheral; ix) External peripherals window.

The simulator main window is shown in Figure 2. The upper part, called the *system window*, includes a bar of pusbuttons used to control system operation. For example, the RESET button is used to reset all the microcontrollers in the system. The external peripherals are also shown in this area, each type with its own picture. Their connections are depicted using node numbers or labels.

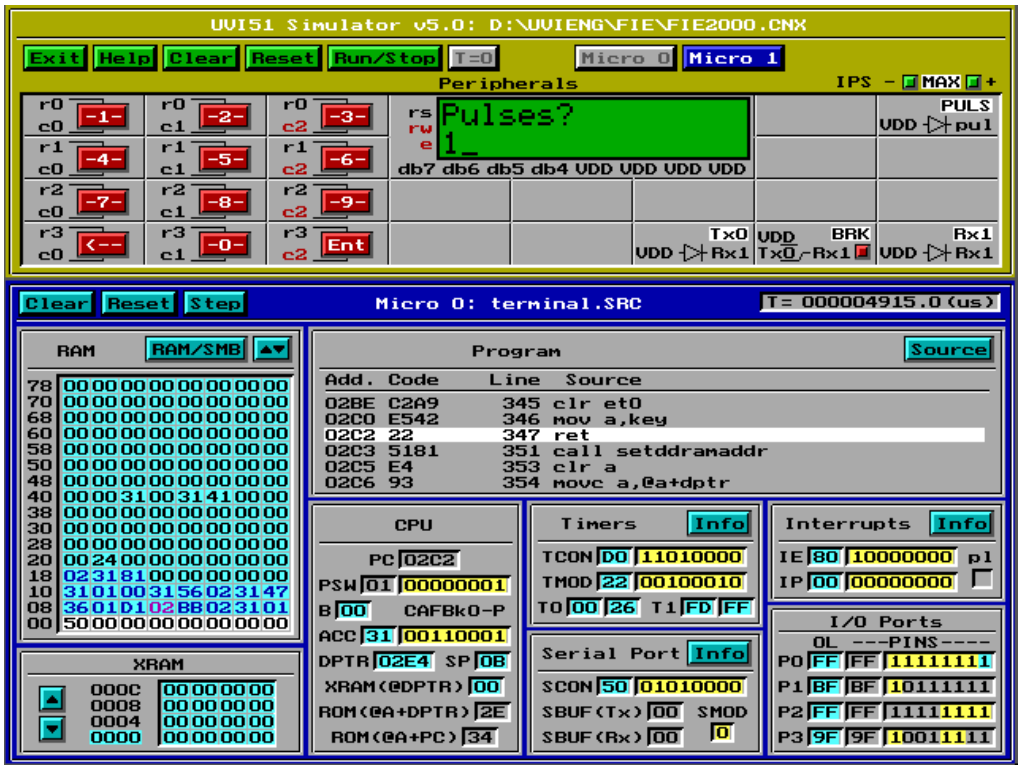

FIGURE. 2
SIMULATOR MAIN WINDOW.

The lower part of the main window is called the *microcontroller window*. Even though the tool can simulate up to 4 microcontrollers concurrently, only one can be displayed at a time. A bar of buttons is used to control the simulation of the displayed microcontroller. For example, the RESET button is used to reset only the displayed microcontroller and the STEP button is used to execute one instruction of the displayed microcontroller. The other microcontrollers in the system will execute the number of instructions needed to keep synchronism.

All the embedded peripherals are simulated. The microcontroller window includes a set of subwindows that show the current state of a part of the microcontroller: the CPU core, internal and external RAM, timers/counters, etc. In addition, a program window is used to show the code that is being executed.

Additional windows depicting further details of each embedded peripheral are available. These windows can be opened by clicking the corresponding *Info* buttons. The state of the peripherals is shown using a graphical style similar to that used in manufacturers' handbooks. This ensures a fast recognition by the students. When in single step mode, these windows are interactive, so that the user can change control bits and see the effects of such changes on the peripheral configuration. As an example, the timers/counters graphic window is shown in Figure 3.

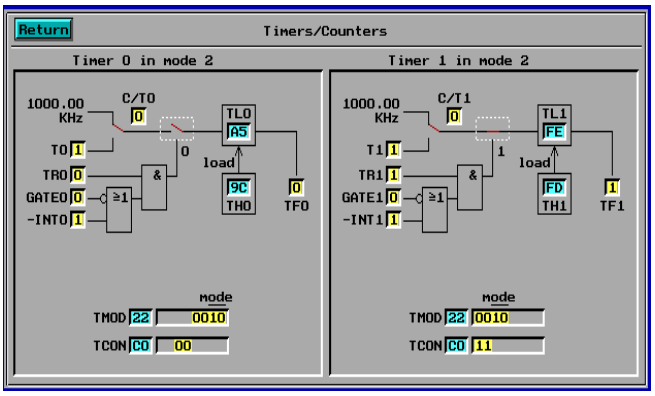

FIGURE. 3
TIMERS/COUNTERS GRAPHIC WINDOW.

Special attention has been paid to achieve a very realistic simulation of the serial port operation, but only modes 1 and 3 can be simulated in the current version of the tool. Serial port simulation is performed at logic level in the TxD and RxD signals, with the real timing given by the baud rate that the timer 1 provides. In this way, a simulation of a serial port connected to a software UART can be performed. The 8051 serial port multiprocessor mode can also be simulated. The serial port graphic window, shown in Figure 4, indicates the current mode, baud rate, and the transmitting and receiving state.

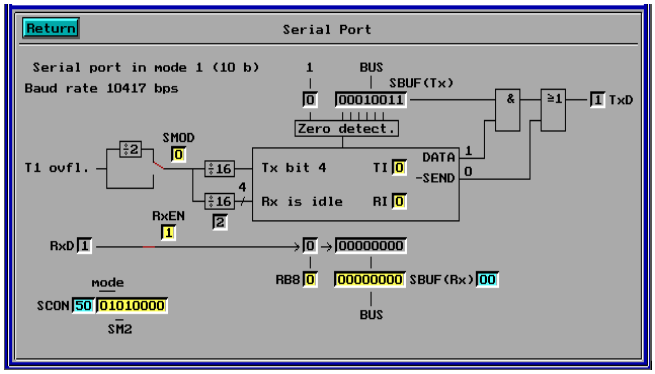

FIGURE. 4
SERIAL PORT GRAPHIC WINDOW.

THE LOGIC ANALYZER

If an external peripheral of type logic analyzer has been defined in the system configuration file, the simulator writes, when running, the chronograms of every node in the system to a file. Once the simulation is terminated, the user can inspect the chronograms by invoking the logic analyzer from the assistant.

All the nodes in the system are available for inspection. Chronograms can also be shown in the form of buses. Time scales and a time cursor are included to allow time measurements. The cursor can also be used as a reference to align chronograms. *Zoom in* / *zoom out* and *find pattern* utilities are also available. Figure 5 shows the logic analyzer window.

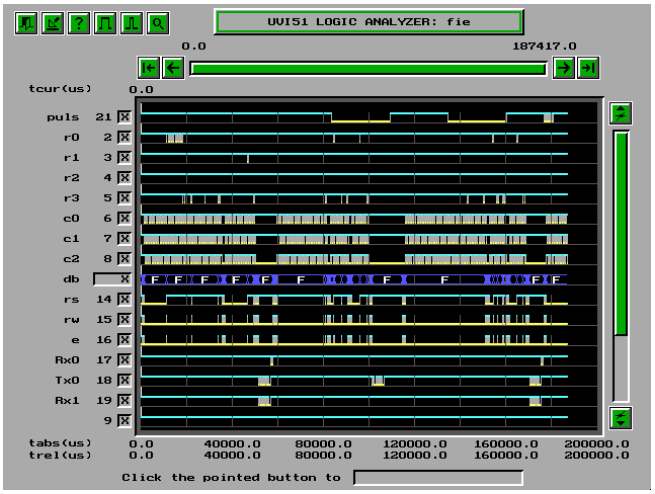

FIGURE. 5
LOGIC ANALYZER WINDOW.

AN APPLICATION EXAMPLE

This section describes an intermediate level example to be developed by the students. This example is specified in the following terms.

"Design a system consisting of two 8051 microcontrollers interconnected via an asynchronous serial link, intended to generate pulse bursts. The first microcontroller, called *terminal* must be connected to a 12 key matrix keyboard and to a standard 2 lines by 16 characters LCD screen. When *terminal* starts from reset, it has to show in the LCD screen the message "Pulses?" and to wait for the user to introduce from the keyboard a number between 1 and 127. Then, *terminal* has to show the message "Period?" so that the user can introduce pulse duration in tenths of miliseconds in the range 1 to 127. When both parameters have been introduced, *terminal* has to send their values to the second microcontroller, called *pulse generator.*

Pulse generator must generate a pulse burst in response to each message received via the serial link. The number of pulses and their duration must agree with the received parameters. The idle level of the output pulses must be high. The time between consecutive pulses of a same burst must be equal to pulse duration. Moreover, *pulse generator* must answer either with an acknowledge (ACK) or not acknowledge (NAK) message to *terminal* before the burst starts. The NAK message indicates a bad format or a number out of range.

Both microcontrollers must operate at 12 MHz. The serial link must operate at 10417 bits per second with an 8-bit data format and without a parity bit. Messages are to be sent as ascii codes."

The block diagram in Figure 6 shows the system hardware structure. The matrix keyboard is connected to the *terminal* P1 port. The LCD screen is connected using a 4-bit data bus and the 3 standard control signals RS, RW and E.

The *terminal* TxD and RxD pins are connected to the *pulse generator* RxD and TxD pins. In a real application these connections could be made through RS232C buffers, an ASK radio link or any other suitable channel. In order to test system response to an eventual link fail, the simulation must provide a way to open the link between the *terminal* TxD pin and the *pulse generator* RxD pin. This is the reason why the BRK switch is included.

The students begin their work writing the suitable configuration file.

The system configuration file asigns a program called terminal.src to the first microcontroller. The second one is assigned a program called pulsegen.src. All the external peripherals are defined: Twelve pushbuttons (keyboard), one LCD screen, three LEDs, and a switch. The external peripherals are shown by the simulator as depicted in Figure 7.

The next step is to write the source code file for each microcontroller. The best option is to start writing the code for *terminal*, and debug it until the right signals are obtained in its TxD pin. The students can verify all the signals using the logic analyzer, as shown in Figure 8. These include keyboard scan signals and the LCD screen interface.

Once *terminal* is operating properly, the students write the source code file called pulsegen.src. This program can be easily debugged using the keyboard conected to *terminal* to send the convenient requests.

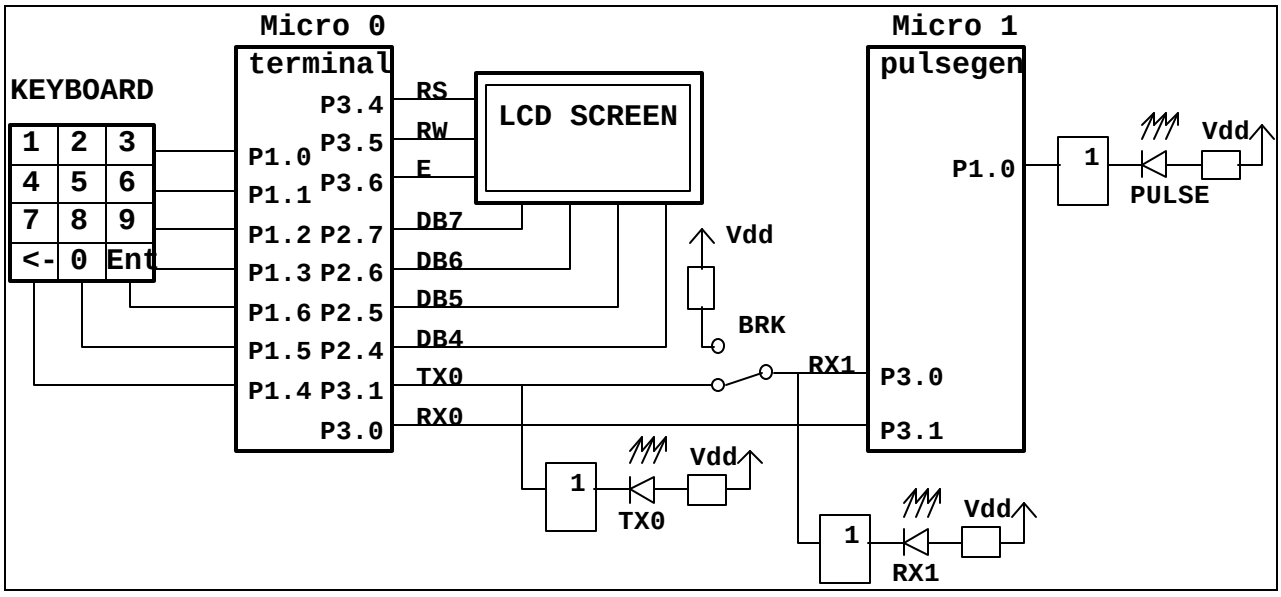

FIGURE. 6
BLOCK DIAGRAM FOR THE PROPOSED EXAMPLE.

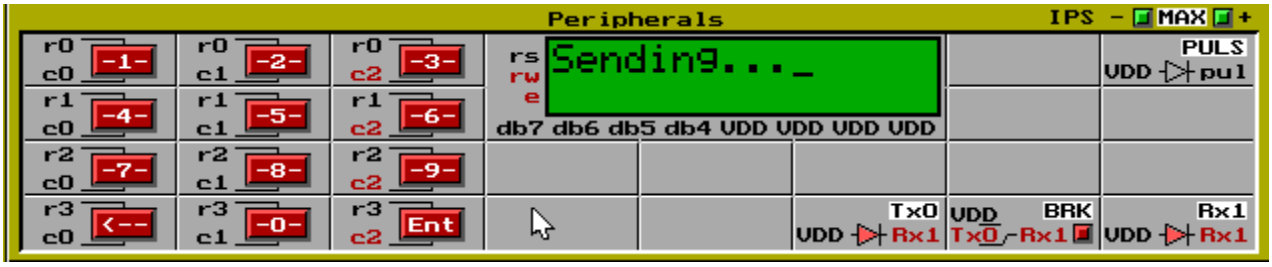

FIGURE. 7
SIMULATOR PERIPHERALS WINDOW FOR THE PROPOSED EXAMPLE.

Finally, the switch placed between *terminal*'s TxD and *pulse generator*'s RxD can be left open before a request is sent by *terminal*. Since *pulse generator* does not receive any request, it does not generate any pulse and does not answer to terminal. *Terminal* must detect this situation as a time-out and show a suitable message in the LCD screen.

Once the whole system is operative, students can introduce new specifications or refinements in order to learn more about keyborad scanning, LCD screen control, serial links, use of timers, serial ports, interrupts, etc.

CONCLUSION

A software tool for teaching/learning the 8051 microcontroller, called UVI51, has been presented. It can be used in the classroom when describing the microcontroller architecture, the instruction set or the embedded peripherals operation, and also in the laboratory to write and test application programs. A copy of UVI51 can be delivered to the students, so they can make some home-work before they enter the laboratory.

UVI51 is being used in introductory courses on microcontrollers. Students instruction is complemented with the use of a low-cost in-circuit emulator, based on the downloadable flash versions of the 8051 family [8]. The user can request the download process from the UVI51 assistant. The results of the application of our approach to education have been excellent both in terms of student motivation and scores.

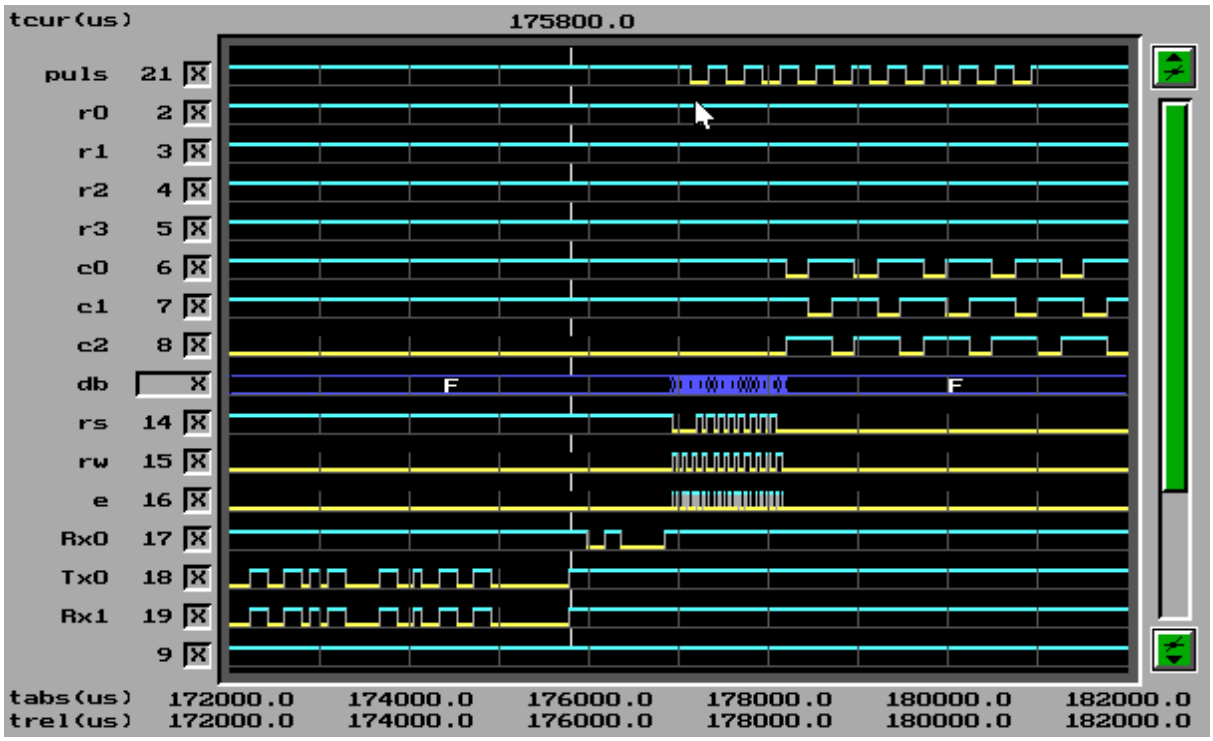

FIGURE. 8
LOGIC ANALYZER CHRONOGRAMS.

The simulator operates at instruction level, but our work is being oriented to develope a new version operating at machine-cycle or clock-cycle level. This will aid in two aditional improvements, at the expense of decreasing performance: i) Timer 2 and new serial port modes simulation for the 8052 microcontroller; ii) External bus detailed simulation. Other possible improvements include undo and save/load state capabilities.

The current version of UVI51 runs under MSDOS and is available, free, in the web site http://www.dte.uvigo.es. In the near future, versions running under Microsoft Windows and LINUX will be developed.

REFERENCES

[1] Nunnally, C.E. , "Teaching Microcontrollers," *Proc. of the 26th Frontiers in Education Annual Conference*, Vol 1, Nov. 1996, pp. 434-436.

[2] Avocet Systems, AVS High-Level Simulator/Debugger, http://www.avocetsystems.com/.

[3] Keil Software, dScope Simulator/Debugger, http://www.keil.com/.

[4] Virtual Micro Design, Universal Microprocessor Program Simulator (UMPS), http://www.vmdesign.com/.

[5] Rodriguez Andina, J.J., del Río, A., "Aplicación de un simulador al desarrollo de prácticas con microcontroladores," *Proc. of the 2nd Conference on Tecnologías Aplicadas a la Enseñanza de la Electrónica*, Vol. III, Sept. 1996, pp. 42-46 (in Spanish).

[6] Intel Corp., *MCS-51 Macro Assembler Manual (V2.2)*. 1986.

[7] IAR Systems, *Micro Series 8051 C-Compiler Manual (V4.10A/DOS)*. 1991.

[8] Atmel Corporation, *8051 Flash Microcontroller Data Book*, Dec. 1997. http://www.atmel.com/atmel/products/prod20.htm.