
A picocontroller training simulator in a Web page

M. Collier

*Department of Electronic Engineering, National University of Science and Technology,
Bulawayo, Zimbabwe*

E-mail: collier1942@yahoo.co.uk

Abstract A software simulator hosted in a Web page is described, and the evaluation of its effectiveness by students of the National University of Science and Technology in Zimbabwe is discussed. The processor which is simulated is the PIC16F84 picocontroller, and the screen display shows the source program, RAM locations and Special Function Registers. To minimise the size of the package, the functionality has been reduced so that it only displays a set of sample programs representing a number of key techniques in the programming of the device. Users can step through programs observing the memory changes to facilitate an understanding of the operations.

Keywords software simulation; web-based training

With the pace of development in electronics, the training and education of engineers demands a steepening of the learning curve if students are to familiarise themselves quickly with new devices. This is particularly true in the case of programmable integrated circuits, where familiarity with the interaction between software instructions and hardware architecture is essential to achieving design expertise. The plethora of new offerings in this field is a stimulus to the introduction of rapid-learning techniques into the classroom environment.

Among the devices introduced in the electronic engineering degree course at the National University of Science and Technology (NUST) in Zimbabwe is the picocontroller. This variant of the microcontroller provides a flexible, cheap processing unit which lies at the heart of many student projects and industrial consultancies in the country. Consequently it is necessary for students to gain a solid grasp of the intricacies of the hardware and the programming techniques as swiftly as possible. Therefore the regular lecture course and laboratory sessions have been complemented by the provision of a training simulator on the departmental Intranet.

The requirements of the simulator were that it should be easy to access, provide several selectable sample programs, and cover a some of the more intricate skills necessary to utilise the chip effectively.

Details of the picocontroller

One of the devices which have been chosen for teaching and project work in the embedded computer systems course at NUST is the PIC16F84,¹ which is an 8-bit microcontroller manufactured by Microchip. This version incorporates a considerable amount of EEPROM program memory, which can be rapidly downloaded and erased. In consequence the design and development cycle can be shortened because

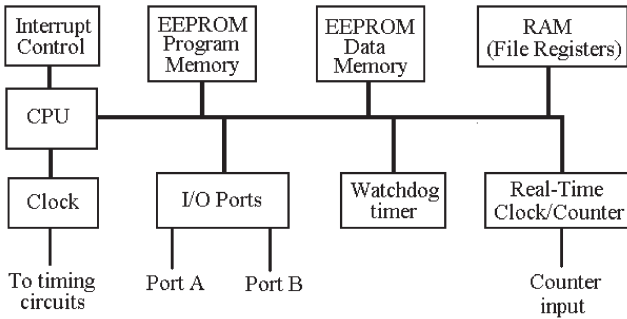


Fig. 1 Architecture of the PIC16F84 picocontroller.

rapid iterations in program development are possible. The basic hardware structure of the PIC16F84 is shown in Fig. 1. Input/output facilities of the device include:–

- 1 One 8-bit bi-directional port.
- 2 One 5-bit bi-directional port, of which one line is open-collector.
- 3 One external interrupt input.
- 4 One counter input.

Versions are available for either 4MHz or 10MHz clock oscillators, which may be implemented using *R-C* circuits or crystals. A reduced instruction set of opcodes provides single-cycle execution of most instructions, resulting in fast performance, which is further enhanced by a two-stage instruction pipeline.

The device incorporates 36 bytes of user RAM, known in Microchip terminology as File Registers. A further 64 bytes of non-volatile data memory is available for storage and retrieval by the program.

In common with most microcontrollers the PIC utilises a bank of Special Function Registers (SFRs) for controlling the hardware parameters. These are summarised in Table 1.

The concept of an embedded applet

The advent of the Internet has made possible access to information in a structured and standardised manner. Within the Electronic Engineering Department at NUST, a local area network based on the Windows-NT operating system provides an Intranet that is available to both students and staff.

One way to implement the software simulator for the picocontroller would be to make it available as a downloadable file from the system server. This would require the user to download the file and then execute it. A more user-friendly approach has been adopted by providing the simulator within an Internet web page, which is ready to run as soon as the page is displayed.

The page contains the simulator in the form of an interactive applet, which accepts

TABLE 1 Special Function Registers

SFR Name	SFR Function
IND0	Indirection data register
RTCC	Real-time clock/counter
PCL	Program counter (low byte)
STATUS	Various status flags
FSR	Indirection address register
PORTA	8-bit bi-directional port
PORTB	5-bit bi-directional port
EEDATA	Non-volatile data register
EEADR	Address register for non-volatile data
PCLATH	Program counter (high byte)
INTCON	Interrupt enable register
OPTION	Timer control register
TRISA	Data direction register
TRISB	Data direction register
EECON1	Non-volatile read/write control
EECON2	Dummy register for non-volatile data
W	Working register

input through mouse clicks on screen buttons, and displays values of the registers within the picocontroller.

Since the access time to the Internet page needs to be kept low, especially if it is being accessed by a remote site on the Internet, the size of the applet code has been reduced to the minimum consistent with the educational objectives of the simulator. As a result the range of activity has been deliberately restricted to allow basic familiarisation with the picocontroller, without providing full functional simulation.

The facilities of the simulator

Upon opening the Web page, the user is presented with the screen display shown in Fig. 2, from which a sample program may be selected. On mouse-clicking the selector box, a menu of programs highlighting various programming techniques is offered, as seen in Fig. 3. Once a program has been chosen, the source code appears in the left-hand window together with a red bullet indicating the next instruction to be executed.

By clicking the 'Step through Program' button each instruction can be executed, while indicating at all times the next instruction by the position of the bullet. The centre window shows the RAM File Registers and their contents at any time, while the right-hand window lists the Special Function Registers, their addresses, and their current contents. Values of all registers are updated at each step, as shown in Fig. 4. Since the input/output lines are treated as SFRs, the state of outputs can be seen from the PORTA and PORTB registers.

The 'Reset' button returns the program counter to zero and the bullet to the first

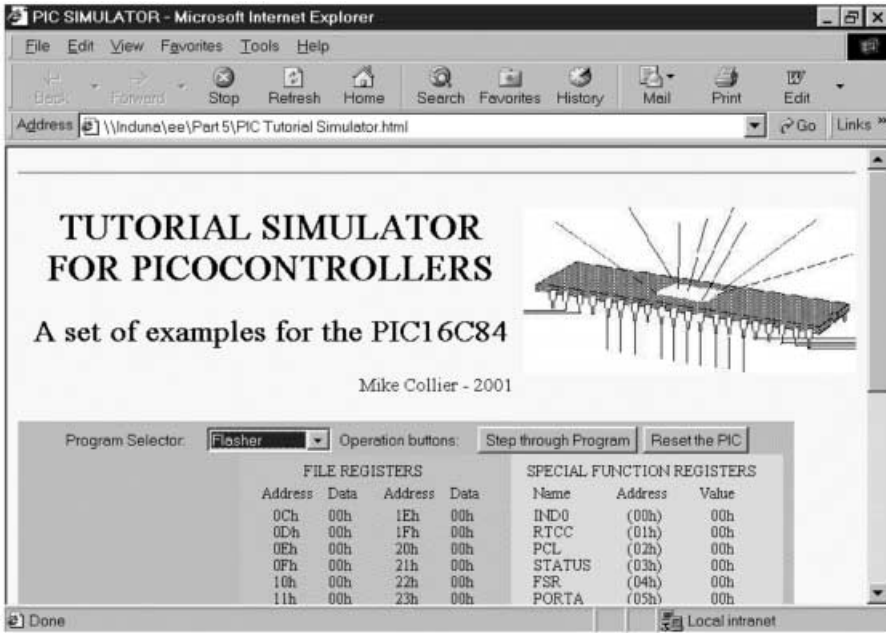


Fig. 2 Opening display of the simulator.

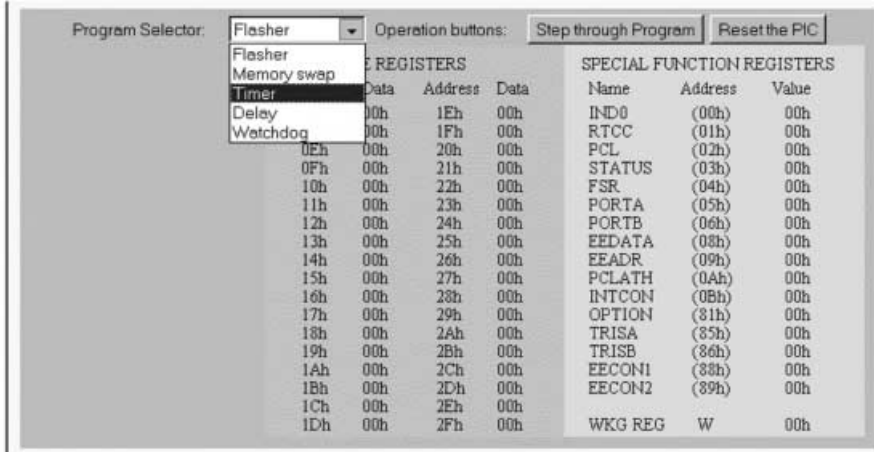


Fig. 3 Selection of example programs from memory.

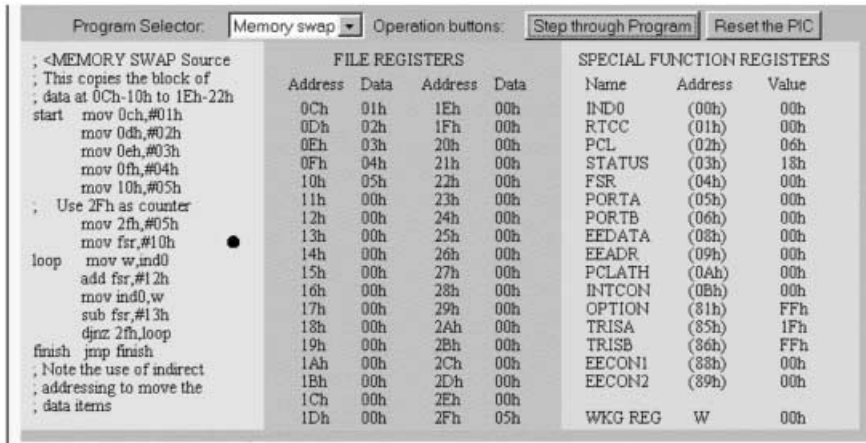


Fig. 4 Stepping through a sample program.

executable instruction (thereby ignoring comments in the program). The File Registers are cleared, and the SFRs are set to the values for Power-On Reset.

The following sample programs are provided by the software:

- 1 *Flasher program* This causes Ports A and B to flash out of phase in an endless loop.
- 2 *Memory swap program* A block of memory is copied to another location, using indirect addressing.
- 3 *Delay program* Three nested loops are used to provide a fixed delay routine.
- 4 *Timer program* The configuration code is executed for setting up the timer in association with the prescaler, and for starting the counting.
- 5 *Watchdog program* A simplified exhibition of the operation of the watchdog timer is presented in which the effect of a closed loop in the program is overcome by the automatic resetting of the device when the watchdog timer overflows.

Detailed source code of these programs in Parallax assembly language is given in Appendix A.

The Java program of the applet

The applet is entirely event-driven, which is achieved by instantiating the standard event handlers as follows:

```

Button stepButton = new Button("Step through Program");
Button resetButton = new Button("Reset the PIC");
Choice menu = new Choice();

```

These are then handled by overriding the standard `action()` method with the following set of conditionals:–

```

Public boolean action(Event evt, Object arg)
{
    if (arg == "Step through Program") step( );
    if (arg == "Reset the PIC") reset( );
    if (arg == "Flasher") loadProgram("Flasher");
    if (arg == "Memory swap") loadProgram("Memory Swap");
    if (arg == "Timer") loadProgram("Timer");
    if (arg == "Delay") loadProgram("Delay");
    if (arg == "Watchdog") loadProgram("Watchdog");
}

```

The program makes use of widgets from the Abstract Windowing Toolkit (AWT) to implement the GUI for both input and output.

The assembler chosen for tuition purposes at NUST is the Parallax assembler, which has a close resemblance to the Intel ASM51 assembler, rather than the Microchip assembler which uses a very different set of conventions. The Java program parses the running program line by line as the program counter changes, and expedites the necessary changes to registers and program counter after each instruction.

Evaluation of the simulator

The target group

The simulator was tested by making it available on the departmental Intranet to students of the final year class, as part of their course in embedded computer systems. To encourage participants to take the package seriously, both for their own benefit and for the purposes of evaluation, a sheet of questions was given to each student and was subsequently marked. In addition an evaluation questionnaire was used to elicit personal comments on the usability and relevance of the package. The test sheet is given in Appendix B, and the questionnaire in Appendix C.

Students' responses

A total of 27 students participated in the survey. They had been give three hours of lectures introducing the picocontroller, together with a published set of lecture notes.² Following this they were allowed access to the simulator on the departmental network, and given a week in which to complete the test and questionnaire.

The following responses were received:

- 1 The average time taken to complete the test was 89 minutes, with 30 and 180 minutes being the fastest and slowest respectively.
- 2 The following features were found to be helpful:
 - the flow indicator bullet showing the next instruction to be processed
 - the variety of programs
 - the facility to step through the programs

- the reset facility
 - the use of the indexed addressing mode
- 3 Suggestions for further sample programs which could usefully be included were:
 - serial communication
 - interrupt handling
 - 7-segment decoding
 - counting of external events
 - stepper-motor control
 - 4 The average students' rating of the usefulness of the simulator to reinforce one's understanding of the PIC16C84 was 4.44 (out of a maximum of 5).
 - 5 The average students' rating of the ease of use of the package was 4.96.
 - 6 The average students' rating of the relevance of the simulator to the material of the lecture course was 4.74.
 - 7 Suggestions for further improvement included:
 - highlighting of registers when they change value.
 - use of the Enter key for 'step-through-program', as well as mouse-clicked button.
 - provision of Help facilities.
 - binary representation of bit-addressable Special Function Registers.
 - a further simulator for the Intel 8051 series of microcontrollers.
 - an option for students to enter their own programs for simulation.

Results of the evaluation

It is concluded that students had no difficulty with the controls of the simulator, and that it was generally found to be user-friendly. They acknowledged the relevance of the software to the material of the embedded computer systems course, and felt that it enabled learning of basic programming techniques in a short time.

There was general acceptance of the screen display and colour choices, with positive comments about the clarity of presentation. Suggestions were made for highlighting memory as it changed and for increasing the range of stepping controls.

There was general interest in the range of example programs provided, but several valuable suggestions were made for additional programming topics.

Conclusion

The concept of a simple training tool for familiarisation with the elements of a picocontroller has been described, and encouraging student evaluation has been reported.

The enormous boost given to computer-based learning by the introduction of the Internet means that students now expect to be able to access necessary information quickly and conveniently. As a result the idea of a simulator embedded in a web page has great attractiveness. Students can rapidly evaluate the functions and interfaces of the electronic device, thereby giving them confidence in approaching the real hardware, an opportunity to analyse programs in detail, and an understanding of the limitations of what they are using.

References

- 1 PIC16/17 Microcontroller Data Book (Microchip, 2000).
- 2 M. Collier, An Introduction to Microcontrollers and Picocontrollers, NUST Lecture Notes Series No.6, NUST, 1999.

Appendix A Source programs of the samples

1 Flasher program

```

;(FLASHER) Source Program
; This alternately flashes
; Ports A and B.
start  mov PortA,#00h
        mov PortB,#0ffh
; Several nops for delay
        nop
        nop
        nop
        mov PortA,#0ffh
        mov PortB,#00h
        nop
        nop
        nop
        jmp start
; Loop round indefinitely

```

2 Memory swap program

```

;<MEMORY SWAP Source
; This copies the block of
; data at 0Ch-10h to 1Eh-22h
start  mov 0ch,#01h
        mov 0dh,#02h
        mov 0eh,#03h
        mov 0fh,#04h
        mov 10h,#05h
; Use 2Fh as counter
        mov 2fh,#05h
        mov fsr,#10h
loop   mov w,ind0
        add fsr,#12h
        mov ind0,w
        sub fsr,#13h
        djnz 2fh,loop
finish jmp finish
; Note the use of indirect

```

```

; addressing to move the
; data items

```

3 Delay program

```

; <DELAY> Source Program
; Illustrates the use of
; 3 loops to produce delay
start  mov 11h,#03h
loop1  mov 12h,#02h
loop2  mov 13h,#05h
loop3  djnz 13h,loop3
       djnz 12h,loop2
       djnz 11h,loop1
; End of delay routine
finish jmp finish

```

4 Timer program

```

; <TIMER> Source Program
; Code for starting Timer
; to count 65336 machine
; cycles before overflow
start  setb gie
       clrb rtie
       clrb rtif
       mov rtcc,#00h
; Set prescaler
       setb rp0
       clrb rts
       clrb psa
       or option,#07h
       clrb rp0
; Start timer running
       setb rtie
; Enter Main Program
finish nop
       jmp finish

```

5 Watchdog program

```

; <WATCHDOG> Source
; For illustration the WDT
; is assumed to overflow after
; 10 cycles (instead of 18 ms)
start  clr wdt
       mov 20h,#04h
loop1  mov 21h,#05h

```

```
; A programming error in the
; next line causes endless
; looping
loop2 jmp loop2
      djnz 20h,loop1
      clr wdt
finish jmp finish
; As long as the PIC clears
; the WDT within 10 cycles, the
; execution will continue.
; When it gets stuck inside the
; loop the WDT resets the device
```

Appendix B Questions in the student test

Flasher program

- 1 Which registers, if any, change value when the nop instructions are executed in the program?
- 2 What are the values of Port A and Port B on reset?
- 3 For how many cycles does the value of Port B remain at FFh?

Memory swap program

- 4 Which particular addressing modes are used in this program?
- 5 What is the significance of the numbers 12h and 13h used in the program?
- 6 What are the values found in the following just after the value in register 2Fh becomes 02h? Registers 0Ch to 10h; Registers 1Eh to 22h; FSR; INDO; W; PCL
- 7 Why is indirect addressing useful in moving large blocks of data in memory?

Delay program

- 8 How many times do the following loops execute? loop1; loop2; loop3
- 9 What is the relation between the immediate values put into the registers and the numbers of times the loops execute?

Timer program

- 10 What does or option,#07h do?
- 11 What does the instruction clrb psa do?
- 12 Why is rp0 set and cleared in the program?

Watchdog program

- 13 What is the highest program memory location reached by the running program?
- 14 Why is the instruction djnz 20h,loop1 not executed?
- 15 What do you think the line with the error should have been?

Appendix C Student evaluation questionnaire

- 1 How long (in minutes) did it take you to complete the technical questions on the test sheet?
- 2 Which of the controls of the simulator did you have difficulty understanding?
- 3 Which features of the simulator did you find particularly helpful?
- 4 Please comment on the clarity of the screen presentation, colour usage and font size.
- 5 Please suggest any other types of PIC programs that you would like added to the simulator.
- 6 Rate the simulator (on a scale of 1 to 5) in terms of its usefulness to reinforce the material of the lecture course. (5 = very useful; 1 = useless)
- 7 Rate the simulator (on a scale of 1 to 5) in terms of its ease of use. (5 = very easy; 1 = very difficult)
- 8 Rate the simulator (on a scale of 1 to 5) in terms of its relevance to the material of the lecture course. (5 = very relevant; 1 = irrelevant)
- 9 Any other comments you would like to make to the design team.