An Educational Board for Teaching Microcontroller System Design Laboratory
Abdallah Kassem, Mustapha Hamad, Charly Bechara and Manar Khattar
Pages 1 – 9
DOI: http://dx.doi.org/10.21700/ijcis.2016.101

An Educational Board for Teaching Microcontroller System Design Laboratory

Abdallah Kassem, Mustapha Hamad, Charly Bechara and Manar Khattar
Faculty of Engineering, ECCE Department
Notre Dame University-Louaize
Zouk Mosbeh, Beirut, Lebanon
[akassem, mhamad, cebechara, mfkhatar]@ndu.edu.lb

Abstract: *Teaching undergraduate electrical and computer engineering students microcontroller/microprocessor design concepts is essential in today's highly advanced technological environment. Furthermore, we believe that all engineering students should take an introductory course in this area. With this objective in mind, we have developed an educational board to teach microcontroller circuit design laboratory. The development board is based on PIC16F877A microcontroller from Microchip. In addition, essential software routines, and a laboratory manual that contains a list of design-applications experiments are part of this lab.*

Keywords: *Microcontroller, Engineering Education, Development Board, Data Acquisition, Laboratory, PIC, I2C, IR, EEPROM.*

Received: May 01, 2016 | Revised: August 10, 2016 | Accepted: September 14, 2016

1. Introduction

As technology advances, curriculum and laboratories are challenged to keep pace [1-7]. This is particularly accurate in electrical and computer engineering, where the range of technologies is constantly increasing and diversifying. Currently, the ECCE department at Notre Dame University (NDU-Lebanon) offers two courses and one laboratory focusing on embedded systems: EEN 324, EEN 325 (microprocessor design and its associated lab) and EEN 326 (introduction to microcontrollers). Advanced topics in this area are usually delayed until the graduate level, and thus causing undergraduates to lack some design training. In addition, due to the increasing importance of system level design methodologies, aspects of hardware-software codesign should be incorporated in the curriculum [8]. In fact, many books [9, 10] have been written targeting undergraduate system-level embedded design courses. Therefore, we realized that there is a gap between the graduate level codesign course and the undergraduate introductory course. For this reason, we developed a PIC-based microcontroller design laboratory aiming at giving students practical design problems to prepare them for the real world environment. PIC® microcontrollers from Microchip are becoming very popular for developing low-cost and reliable mechatronic systems. The 8-bit microcontroller has enjoyed a tremendous growth in embedded systems applications. It is a self-contained computer-on-a-chip that integrates a microprocessor, ROM and RAM memories, I/O ports and special hardware features.

Microchip's PIC MCU was selected to be taught in this laboratory because it is versatile in the local market, inexpensive and plenty of application notes are available on the Microchip website. Among all the PIC MCU families, the popular PIC16F877A [11] was specifically chosen because of its variety of hardware modules needed for most applications. Also, the 'F' or Flash memory type of PICmicro® microcontrollers is suitable for testing and building programs as they are reprogrammable.

The development board consists of a variety of peripheral devices that can be interfaced with PIC16F877A and that are chosen according to their high usability in students' projects and to their relevance in showing the commonly utilized internal resources of the PIC16F877A.

The lab manual is a set of various experiments that introduce the peripherals of the board and the corresponding PIC MCU internal resources along with software routines that were explained in details using flowcharts, and that were written in assembly code and compiled by MPLAB®. Also the corresponding hardware configurations for the interface of the peripherals with the PIC16F877A were provided in each section.

This paper provides a description of the complete package designed for the Microcontroller Laboratory

Course MLC. The balance of the paper is organized as follows: Section II is an overview of the hardware resources used in the MLC; section III describes the software resources. The different experiments that form the laboratory manual are presented in section IV. Finally section V provides our conclusion.

2. Hardware Resources

The development board shown in figure 1 consists of a 40-pin ZIF socket where the PIC16F877A can be easily plugged.

Figure 1. Development Board

The peripherals interfaced with the PICmicro include display devices such as a 16*2 LCD and a Dual 7-segment display, an input device which is a 4*4 keypad, Infrared transmitter and receiver for wireless communication, analog input devices such as a temperature sensor, a phototransistor and two potentiometers, special ICs to implement the Master Synchronous Serial Port MSSP module [12], which are a D/A converter and a serial EEPROM, and interface devices based on RS-232 protocol to implement the Universal Synchronous Asynchronous Receiver Transmitter USART module.

The PIC16F877A microcontroller has 5 I/O ports, interfaced with the on-board peripherals in such a way to explore all the modules and internal resources of the PIC MCU. The on-board peripheral devices are summarized in Figure 2 and consist of: Display Devices, Input Device, Sound Device, Infrared (IR) Communication, Data Acquisition System, External EEPROM, PC Interface and Mechatronics Application.

2.1. Display Devices

The board includes two display devices: A common cathode dual 7-Segment display and a 16*2 LCD, multiplexed together on PORTD, which is configured as a general purpose I/O port. Other display features such as indicator LEDs are connected to PORTA and can be used either for display purpose or for the illustration of the Comparator Module output status.

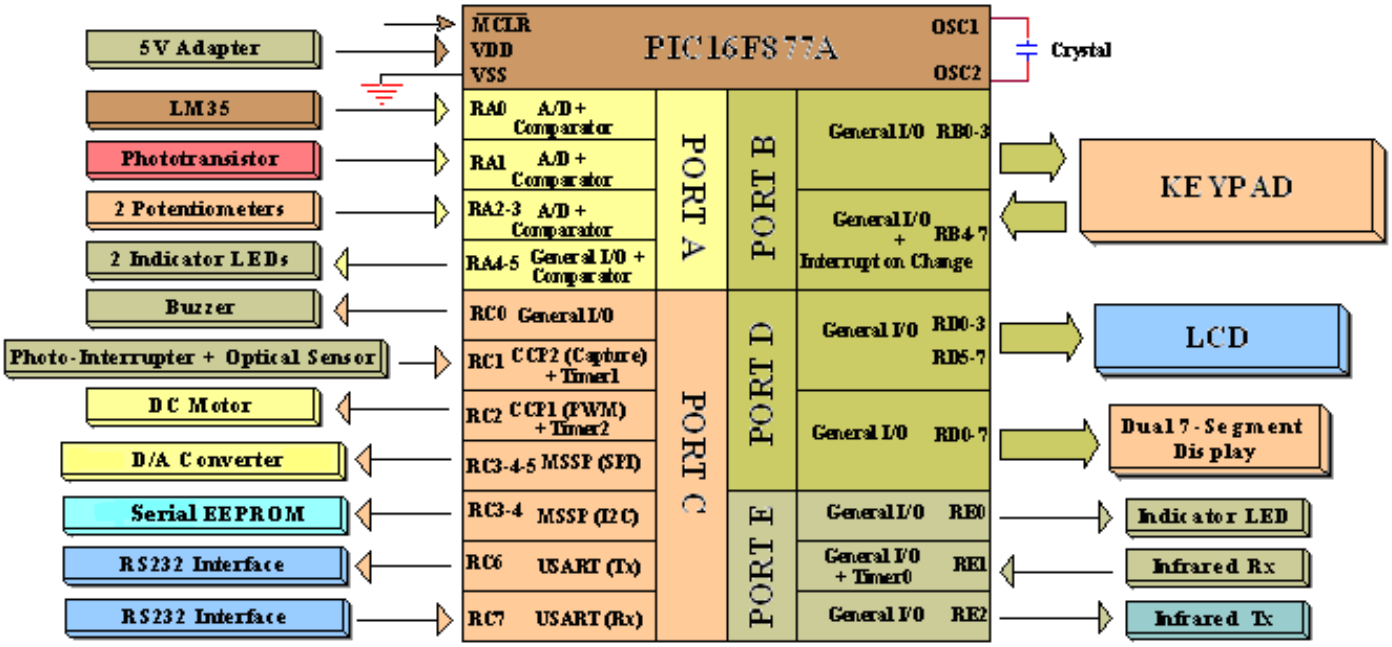

Figure 2. On-Board Peripherals

Because PORTD has only 8 I/O pins and the Dual 7-Segment requires 10, the decimal point (DP) is ignored and the 2 selection lines of the 2 digits are multiplexed on a single pin RD7 through 2 NPN transistors mounted in an inverting configuration. The user can optionally enable or disable any of the 2 digits by a DIP-Switch. The 16*2 LCD based on the popular Hitachi HD44780 screen driver [13] has 14 pins (8 for data, 3 for control and 3 for power), but in order to be fit entirely on PORTD, it was used in the 4-bit mode instead of the 8-bit mode. A 10k potentiometer is placed to adjust the contrast of the panel as shown in figure 3.

2.2. Input Device

The board also includes an input device, which is a 4*4 matrix keypad, the primary input device for real-time systems, found in many electronic devices.

Figure 4 shows the internal connections of the 4x4 matrix keypad. The matrix keypad is well interfaced on PORTB because it has internal pull-up resistors, thus no need for external resistors. In addition, PORTB has special "interrupt on change" pins RB4-RB7 needed for the "Wake Up on Key Stroke" technique, which generates an interrupt when the user hits a button.

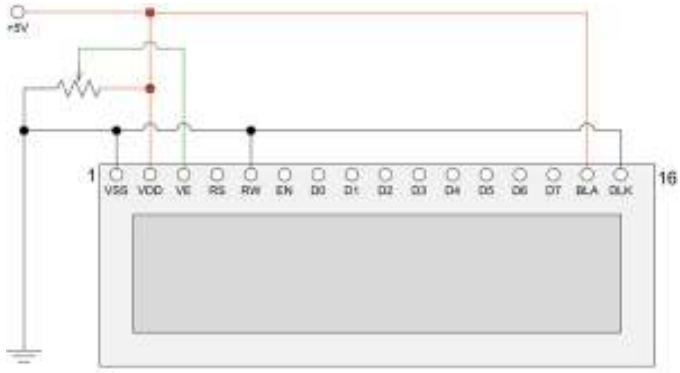

Figure 3. LCD Contrast Adjustments

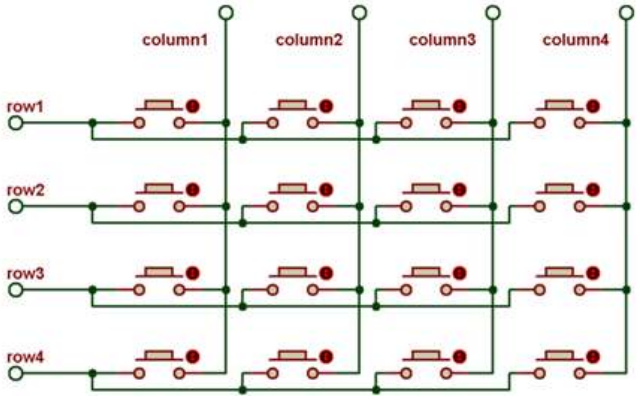

Figure 4. Internal Connections of the 4x4 Keypad

2.3. Sound Device

Sounds are another way of exchanging information; therefore a 5V piezoelectric buzzer was included on the board. It has low power consumption, a negligible electrical noise and a clear penetrating sound. It is interfaced with PIC16F877A on pin RC0 and driven by an NPN transistor to increase the low current supplied by the PICmicro as shown in figure 5.

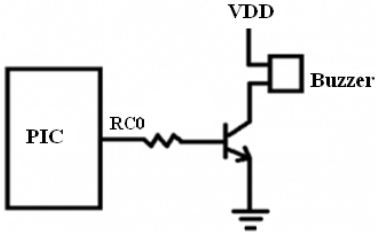

Figure 5. Piezoelectric Buzzer Connection

2.4. Infrared (IR) Communication

The two main devices of IR Communication: Transmitter and Receiver are included on-board. The infrared transmitter is connected on pin RE2 and plays the role of a remote control that can be applied on specific electrical appliances. The PIC16F877A generates the 38 KHz carrier frequency needed to transmit IR signals. On the other hand, the infrared receiver, connected on pin RE1, detects IR signals with 38 KHz carrier frequency and transforms them to electrical signals which are decoded by the PIC MCU.

2.5. Data Acquisition System

The PIC16F877A has an A/D module multiplexed with 8 analog input channels, but it has no internal D/A module. Therefore, an external D/A converter is added in order to complete the Data Acquisition System. The D/A converter communicates with the Serial Peripheral Interface SPI™ bus of the PIC MCU. In the board, 4 channels out of 8 are reserved for 4 analog input devices, which are a temperature sensor, a phototransistor and 2 potentiometers, in order to illustrate the A/D and the Comparator Modules. A jumper connects the analog output of the D/A converter either on-board to a white bright LED or off-board through a SIP connector.

2.6. External EEPROM

Although PIC16F877A has an internal EEPROM, an external serial EEPROM chip is used in order to illustrate the transmission and reception of data through the Inter-Integrated Circuit I2C™ bus. This chip is interfaced on pins RC3 (SCL) and RC4 (SDA), which are specific for I2C serial communication.

2.7. PC Interface

The board is interfaced to the PC through the Universal Synchronous Asynchronous Transmitter Receiver USART module of the PIC16F877A, using a serial cable and the HyperTerminal Software [11]. In this project, the USART is used in the Asynchronous mode of communication because the synchronous mode has been previously described in the SPI and I2C modules, and it is rarely used in USART. The USART is the oldest and still the most predominant interface that implements the asynchronous protocol. The PC is interfaced with PIC MCU using RS-232 protocol. The Serial port connector is the D-Type 9 pins connector. To convert the voltage levels of the PC (RS-232 voltages +12V and -12V) into the corresponding voltage levels of the PIC MCU (TTL voltages 0V and 5V), a level converter chip is used.

2.8. Mechatronics Application

DC motors are used in many types of machinery to perform rotary actions, therefore a complete mechatronic system is built on the board, using both Capture Compare PWM (CCP) modules and using a 5V DC motor and an optical sensor. The DC motor was driven by the PWM (Pulse Width Modulation) mode of CCP1 module, while its speed was measured by an optical sensor using the capture mode of CCP2 module.

3. Software Resources

In addition to the rich development board that consists of different devices for many applications, a

large set of software routines written in assembly code is provided. These routines act as driver programs for each device. A full listing of the software routines and their functions is summarized in Table 1.

Table 1: Summary of Software Routines

Library	Routine name	Routine Function
Delay	DELAY_Short	time delay = [0;771] us
	DELAY_Medium	time delay = [0;196] ms
	DELAY_Long	time delay = [0; 50] s
LCD	LCD_Initialization	Initialize the LCD in 4 bits mode
	LCD_DisplayOff	Set Display Off
	LCD_DisplayOn	Set Display On, Cursor Off
	LCD_CursorBlinkOn	Set Display On, Cursor On, Blink On
	LCD_CursorBlinkOff	Set Display On, Cursor On, Blink Off
	LCD_DisplayClear	Clear entire display and go back to the initial position on the screen
	LCD_ReturnHome	Go back to the initial position on the screen
	LCD_ToLine1	Move to the 1st line, column W
	LCD_ToLine2	Move to the 2nd line, column W
	LCD_SendCommand	Send a Command in 4 bits mode
	LCD_SendCharacter	Send a Character in 4 bits mode
	LCD_SendDigit	Send a Digit in 4 bits mode
	LCD_SendHEX	Display the data with its equivalent HEX number
	LCD_SendBinary	Display the data with its equivalent binary code
	LCD_BackSpace	Move the cursor one character to the left & delete the character
Keypad	KEYPAD_ReadKey	Check which key is pressed on the 4*4 Keypad and returns the corresponding ASCII number
	KEYPAD_Release	Wait till all the keys are released
Infrared Receiver	IR_RX_ReadPacket	Read an Infrared packet from a remote control and saves the command and device ID
Infrared Transmitter	IR_TX_Packet	Transmit an Infrared packet
A/D	AD_SelectChannel	Select the analog channel of the PIC to be sampled
SPI	SPI_SendData	Send a byte using MSSP module in SPI mode
D/A	DAC_SendCommand	Send a Write Command to the D/A converter
I2C	I2C_Start	Initiate a Start Condition
	I2C_Stop	Initiate a Stop Condition
	I2C_Restart	Initiate a Restart Condition
	I2C_ACK	Initiate an Acknowledge ACK Condition
	I2C_NACK	Initiate a Not Acknowledge NACK Condition
	I2C_SendByte	Send a Byte using I^2C bus
	I2C_WaitMSSP	Wait MSSP module to finish its job
	I2C_WaitACK	Wait Slave to acknowledge
EEPROM	EEPROM_Write	Write a byte to the EEPROM at a predefined address (Byte Write)
	EEPROM_Read	Read a byte from the EEPROM at a predefined address (Random Read)
USART	USART_SendCharacter	Send a byte using USART module
Math	UDIV2424L	24 by 24 bit division (microchip website)
	MATH_Convert	Takes a number in binary form and converts it to BCD form

To avoid building a new program from scratch, a sample code is provided. It is divided into 7 main sections:

1. Header of the Program
2. Memory Allocation & Declaration
3. Initialization of Registers and Variables
4. Main Program
5. Interrupt Service Routine
6. Device Routines
7. User Routines

The *Header* of the Program is simply a commented area where the name of the file and the description of the program's function are written. In addition, the processor type and the reset and interrupt vectors are specified in this part.

```
;########################################
;File: Name of the file
;
;Description: description of the program ;
code and its function
;########################################

        LIST    p=16F877A
        include "P16F877A.inc"

;tell assembler what chip is used
;include the defaults for the chip

        ERRORLEVEL      0,      -302
;suppress bank selection messages

        org     0x00    ;Reset Vector
        goto    Main
        org     0x04    ;Interrupt Vector
        goto    ISR
```

The *Memory Allocation & Declaration* section provides a declaration of all the device variables and the specific user variables in a *cblock* memory area. The PORT and TRIS registers are given other aliases according to the name of the connected peripherals. In the following format, the device number is given as an illustration and is replaced by the actual name of the device (i.e: LCD, Keypad, etc…) in the actual sample program. The User Variables area is where the new variables are declared.

```
;=======================================
;     Memory Allocation & Declaration
;=======================================

        cblock  0x20

        ;~~~~~~~~~~~~~~~~~~~~~~~~~~
        ;       DEVICE 1 Variable
        ;~~~~~~~~~~~~~~~~~~~~~~~~~~
        .....

        ;~~~~~~~~~~~~~~~~~~~~~~~~~~
        ;       DEVICE n Variables
        ;~~~~~~~~~~~~~~~~~~~~~~~~~~
        .....

        ;~~~~~~~~~~~~~~~~~~~~~~~~~~
        ;       User Variables
        ;~~~~~~~~~~~~~~~~~~~~~~~~~~
        .....
        endc

DEVICE1_PORT    equ     PORTx
                  ;Device 1 data port
DEVICE1_TRIS    equ     TRISx
                  ;Device 1 port direction
....
DEVICEn_PORT    equ     PORTy
                  ;Device n data port
DEVICEn_TRIS    equ     TRISy
                  ;Device n port direction
```

The *Initialization of Registers* and *Variables* section is where the internal registers of the PIC16F877A and the program variables are initialized. The user can modify or remove some variables depending on his program. For example, if no asynchronous serial communication is required, then the USART registers (TXSTA, RCSTA, BRGH), are omitted.

```
;=======================================
;           INITIALIZATION ROUTINE
;=======================================
Initialize

        ;Disable A/D all digital outputs
        banksel ADCON1
        movlw   b'00000110'
        movwf   ADCON1

        banksel Device1_TRIS
        ....

        ;select Bank0
        bcf     STATUS, RP0
        bcf     STATUS, RP1
        return
```

The *Main Program* section is where the user program starts execution. In the sample program, the *Initialize* routine is the first one to be executed followed by the user code.

```
;=======================================
;               MAIN PROGRAM
;=======================================

Main
        call    Initialize

        ;--> User code starts here

loop    goto    loop
```

The *Interrupt Service Routine* section is where all the interrupts are processed and the corresponding code of each interrupt source is executed. The ISR code starts polling the interrupt source flags according to their priorities and jumps to the routine that matches the source of the interrupt. The user can delete or reassign the priorities of the interrupts according to his need. At the end of each branch of service routines, the *retfie* instruction is executed to exit the ISR and enable further interrupts.

```
;=======================================
;       INTERRUPT SERVICE ROUTINE
;=======================================

ISR
     ;save W, STATUS and PCLATH registers
        movwf   hold_W
        movf    STATUS, w
        movwf   hold_STATUS
        movf    PCLATH, w
        movwf   hold_PCLATH

        btfsc   INTCON, RBIF
        goto    ISR_RBIF
        btfsc   INTCON, TMR0IF
```

```
        goto    ISR_TMR0IF
          .....
        goto    END_ISR

ISR_RBIF
        ;--> User code starts here
        goto    END_ISR

ISR_TMR0IF
        ;--> User code starts here
        goto    END_ISR

END_ISR
   ;reload W, STATUS and PCLATH registers
        movf    hold_STATUS, w
        movwf   STATUS
        movf    hold_PCLATH, w
        movwf   PCLATH
        movf    hold_W, w
        retfie
```

The *Device Routines* section contains all the routines that are summarized in Table1, grouped by Library. Initially, all the software routines exist in the sample program file, but it is the user responsibility to use them correctly according to their given description, and to delete the libraries that are not needed for his specific design in order to benefit from the program memory space.

The *Device Routines* section contains all the routines that are summarized in Table1, grouped by Library. Initially, all the software routines exist in the sample program file, but it is the user responsibility to use them correctly according to their given description, and to delete the libraries that are not needed for his specific design in order to benefit from the program memory space.

```
;=========================================
;               DEVICE 1 ROUTINES
;=========================================

;-----------------------------------------
;NAME     :Name of the routine DEVICE1_Function
;FUNCTION :description of this routine
;INPUTS   :Variables needed to be input before the routine
is called
;OUTPUTS  :Variables output by the routine
;local Variables: Locally used variables
;-----------------------------------------

DEVICE1_Function
        ...
        return
```

Finally, the *User Routines* section is only reserved for any addition software routines built by the user.

The significance of this coding structure is that all the software routines are open-source and accessible by any user. In addition, they are well commented and written in a modular fashion to provide the student with a better understanding of the program's structure and device operation.

4. Lab Experiments

The purpose of the microcontroller laboratory is to teach the students how to build real projects and learn how to interface the mostly used components in their designs.

The laboratory experiments introduced have 5 main objectives:

- Teach all the important internal resources of the PIC16F877A
- Introduce new devices and components useful for any senior project design
- Teach how to interface these devices to the PIC16F877A
- Develop assembly codes and routines useful for controlling the operation of these devices
- Apply these routines to a real experiment project

At the end of the semester, all the ideas and techniques developed in the lab must be implemented in a group work project. This project can be either based on the components of the development board or can be built on any other board using new components.

Upon the successful completion of all the lab experiments and the design project, students will be able to extend their senior projects to a higher level of complexity, since they have acquired most of the basic framework during the lab.

The following is a description of the lab experiments and the benefits that students will gain from each lab unit.

4.1. Experiment 1: Countdown Timer

The purpose of this experiment is to teach how to interface the dual 7-segment display using a technique called multiplexing and to generate a sound by the piezoelectric buzzer using Delay routines provided in the Software section.

This experiment consists of a countdown from 30 to 00 with a decrement each 1 second. After each decrement of the timer, a tick sound is generated by the piezoelectric buzzer driven by a square signal with a fixed frequency. When the timer reaches 00, an alarm sound is generated by the buzzer driven by a constant voltage this time, to alert that the countdown is finished and the process restarts again.

4.2. Experiment 2: Character Generator Display

This experiment teaches the students to interface the LCD with the PIC16F877A. Many LCD routines are provided to help students to initialize and display the desired characters and words correctly. In particular, a program that displays all the characters stored in the CGROM (Character Generator Read Only Memory) table of the LCD with their corresponding Hexadecimal and Binary values will be built. Using the Delay routines, a time delay separates between each two consecutive displays to allow the user to see them clearly on the LCD.

The LCD and the software routines provided in this experiment will be used in all the subsequent

experiments so that the student will be more familiar with this display device.

4.3. Experiment 3: Menu Switching

This experiment introduces the 4*4 Matrix Keypad, which is the major input device in most of the real-time projects. Instead of the traditional polling and scanning techniques of the keypad, the "Wake up on Key stroke" technique is used. It uses the special PORTB<4:7> interrupt on change pins with RBIE interrupt mask. In this case, when a user hits a key, a logic level change on one of PORTB<4:7> input pins occurs, resulting in an interrupt of the running code.

In most user interface projects, several menus exist and the user selects one of these available menus to be displayed on the LCD . In this experiment, the technique of menu switching will be introduced. Assume that key '#' on the keypad is an Enter or Acknowledge key and the key '*' is a Backspace key. The program will read a key from the 4*4 keypad (by interrupt) and display the corresponding menu on a 16*2 LCD (4-bit mode). The user must acknowledge his choice by pressing on key '#' and he may move backward to delete his choice by pressing on key '*'.

4.4. Experiment 4: Infrared Remote Control Transmitter and Receiver

In this interesting experiment, we'll build a complete Infrared communication system based on a specific remote control protocol.

4.4.1. Remote Control Receiver:

After investigating the Infrared communication protocol, the necessary assembly code routines for the detection of the IR packet sent by the remote control were built; these routines will be used to read the IR packet and display its command and device bytes on the 16*2 LCD. Since each button on the remote control has a unique command ID, the student can easily build a control environment using the remote control

The protocol uses the Pulse Code Modulation technique, where logic 1 and logic 0 have different pulse lengths. To measure the received pulse length, Timer0 is configured in timer mode.

4.4.2. Remote Control Transmitter:

After recording each button's command ID in the previous experiment, a remote control transmitter will be built based on these recordings.

This experiment uses the Keypad buttons as the remote control keys. The program waits for the user to hit any button on the keypad. When a button is pressed, the program enters an Interrupt Service Routine and transmits the corresponding Infrared packet of the button. The button pressed is displayed on the LCD.

4.5. Experiment 5: Temperature Sensor

So far, we were dealing with digital inputs and outputs of the PIC MCU. In this experiment, the PIC MCU is interfaced with an analog input device, which is the temperature sensor.

Since the A/D and Comparator modules of the PIC16F877A are multiplexed on the same pins PORTA<0:3>, both modules will be explored in two different experiments.

4.5.1. Heat Alarm System:

For the Heat Alarm System experiment, the internal Comparator module of the PIC16F877A will be explored. The temperature of the sensor is compared to a preset value by the potentiometer, and sounds an alarm by the buzzer if it exceeds this value.

4.5.2. Digital Thermometer:

In the previous experiment, two analog values were compared without knowing the exact value of the temperature sensor. In this experiment, the temperature sensor and the potentiometer values are sampled by the A/D module and then compared by software. Finally, the ambient temperature value is displayed in degree Celsius on the LCD.

4.6. Experiment 6: Speed Measurement of the DC Motor

The speed measurement of a brushed DC motor is the most sophisticated and important experiment. Four internal resources are used in the same experiment: CCP1 (in PWM mode), CCP2 (in Capture mode), Timer1 and Timer2.

The speed of the DC motor will be controlled using CCP1 in PWM mode and measured using CCP2 in Capture mode. It's a closed loop system with the photo-interrupter as the feedback sensor element.

4.7. Experiment 7: Manual Light Control + Optical Receiver

In this experiment, a feedback system that allows the user to control the intensity of light manually is constructed. In fact, the intensity of light is controlled by the digital data bits of the write command register of the D/A converter. These data bits are entered by the keypad. The PIC16F877A sends the input data to the DAC using the Serial Peripheral Interface SPI bus which is integrated in the MSSP module. A white bright LED is connected to the analog output of the DAC. The intensity of the LED varies according to the digital data bits sent by the PIC MCU. This intensity is detected by an NPN Phototransistor, which is connected to the analog pin RA1/AN1 and sampled by the internal A/D converter of the PIC16F877A. The sampled value is displayed on the 16*2 LCD together with the input data from the Keypad. It is a Manual Light Control but it can be extended to be Automatic by removing the Keypad and letting the PIC MCU control the intensity of light according to predefined values.

4.8. Experiment 8: Communication with Serial EEPROM

The previous experiment introduced the SPI bus. The 2nd type of synchronous serial communication handled by the MSSP module is the I2C protocol. In this experiment, we'll communicate with an external EEPROM using the I2C bus by sending a byte to a predefined address then reading it again. The purpose is to test the I2C routines and to introduce the concept of data loggers such as temperature logging, Infrared remote control codes logging, etc…

4.9. Experiment 9: PIC Security Lock

One of the most interesting experiments for engineering students is the communication between the PC and their own system board. The purpose of this experiment is to show how exchanging data is done between the PIC16F877A (using the USART module) and the PC (using the serial port COM1).

The "PIC security lock" experiment demonstrates this idea. The administrator is the user that is sitting in front of the PC and the target device is the PIC16F877A. Initially, the PIC MCU is locked and the administrator should send the correct sequence of digits via the serial port in order to unlock the PIC MCU. The secret code is composed of 3 digits. If the administrator enters the correct code, the PIC MCU transmits the message "correct" to the PC, else it transmits the message "reject".

The user can utilize the HyperTerminal software provided with Windows platform or any other software that emulates the serial port.

5. Conclusion and future work

As part of our on-going continuous improvement in engineering education, and in an effort to keep our ECCE curriculum at NDU-Louaize up to date, we developed a PIC-based educational board for teaching microcontroller circuit design laboratory. We believe that this low-cost and reliable laboratory constitutes, for undergraduate students, a smooth transition to the graduate-level hardware-software co-design courses.
As Future work, this PIC-based educational board can use other programming techniques such as FlowCode or PIC-C and can be connected online to make virtual experiments.

References

[1] M. Hamad, A. Kassem, R. A. Jabr, C. Bechara and M. Khattar, "A PIC-Based Microcontroller Design Laboratory", Proceedings of the 6th IEEE International Workshop on SoC for Real-Time Applications, pp. 66-69, 2006.

[2] Y. Mohanna, M. Hamad, R. Jabr, A. Alaeddine, and O. Bazzi, "Teaching microprocessors, microcontrollers, and digital signal processing courses using only one target processor: the newborn dsPIC30F from Microchip," Computer Applications in Engineering Education, vol. 15, no. 2, pp. 185-191, 2007.

[3] Paul E Morton, "A Student Constructed Microprocessor Development Board for Teaching Microcontrollers", Proceedings of the 2015 ASEE Gulf-Southwest Annual Conference, pp. 1-7, 2015.

[4] C Martinez and SR Pandian, "A Low-Cost Microcontroller-Based System for Computer Interfacing and Control System Education", Proceedings of the ASEE Gulf-Southwest Annual Conference University of Houston, 2011.

[5] A. Suliman, and S. Nazeri, "The effectiveness of teaching and learning programming using Embedded System," 2012 8th International Conference on Information Science and Digital Content Technology (ICIDT), vol.1, pp.32-36, 2012.

[6] (2006) The microchip website. Online. Available: http://www.microchip.com/

[7] Aaron Striegel and Dian T. Rover, "Enhancing Student Learning in an Introductory Embedded Systems Laboratory," 32nd ASEE/IEEE Frontiers in Education Conference, Boston, MA, November 2002.

[8] Joseph Schneider et al., "A Platform FPGA-based Hardware-Software Undergraduate Laboratory," Proc. of the 2005 IEEE Int'l Conference on Microelectronic Systems Education (MSE'05).

[9] Staunstrup and W. Wolf, editors, Hardware/Software Co-design: Principles and Practice, Kluwer Academic Publishers, 1997.

[10] F. Vahid and T. Givargis, Embedded System Design: A Unified Hardware/Software Introduction, John Wiley & Sons, 2002.

[11] 28/40 pin Enhanced FLASH Microcontrollers (PIC16F87XA), Microchip

[12] Technology, Inc., 2001, doc.no. DS39582A.

[13] R.L. Stevens, Serial Communications. Square 1 publication, 2002Dot Matrix Liquid Crystal Display Controller/Driver (HD44780U), Hitachi, 1999.

Dr. Abdallah Kassem received his B.S in Microelectronics from University of Quebec in Montreal in 1992, his M.Sc and Ph.D in microelectronic from Ecole polytechnique de Montreal in 1996 and 2004 respectively. From 1996 to 2000, he taught computer architecture, microprocessors and digital electronic courses and laboratories at AUB, LAU in Lebanon. Dr. Kassem joined Notre Dame University in Lebanon as an assistant professor in the Electrical and Computer Engineering department. He was promoted to the rank of associate professor in 2011. His research interests

include microelectronics design and test, VLSI, semiconductor device modelling and simulation, microprocessors, engineering education/ management and ultrasonic applications.

Prof. Mustapha Hamad earned a BS degree and an MS degree in electrical engineering in 1989 and 1991 respectively, and a Ph.D. degree in microelectronics in 1995 all from the University of South Florida, Tampa. He joined the faculty of engineering at Notre Dame University in October 1997 as an assistant professor in the department of Electrical Computer and Communication Engineering (ECCE). His research interests include microelectronics design and test, VLSI, semiconductor device modeling and simulation, and engineering education. Currently, Dr. Hamad teaches courses in the area of electronics (analog and digital) and is the coordinator for several fundamental courses in the ECCE department. He authored and co-authored a number of articles published in refereed journals and conferences.

Charly Bechara received his Bachelor of Engineering degree in computer communication from Notre Dame University-Louaize, Lebanon in 2006, and his MS degree in System-On-Chip design in 2007 from KTH, and his industrial Ph.D. in computer engineering from Universite Paris Sud in 2011. He is currently an R&D engineer at Tredzone, France.

Manar Khattar received her Bachelor of Engineering degree in computer communication from Notre Dame University-Louaize, Lebanon in 2006, and her MS degree in Electrical and computer engineering from the American University of Beirut in 2008.