Practical Activity #2
IO Configuration and Control

Exercise Objectives:
1. Configure the general purpose I/O ports of the PCI16F877A
2. Interface LCD and numeric keypad
3. Developing basic embedded systems application utilizing LCD and keypad

Student Outcomes:
At the end of the exercise, students must be able to:
1. understand how to configure the GPIO ports of the PIC16F877A MCU
2. able to interface parallel IO devices the LCD and numeric keypad;
3. develop an application utilizing the LCD and keypad.

Tools Required:
The following will be provided for the students:
1. MPLAB IDE v8.92 + MPLAB XC8 v1.33
2. Proteus v8.11

Delivery Process:
The instructor will conduct a review on LCD and merit keypad control as well as a demo on GPIO configuration. All inquiries pertaining to the latter must be referred to this manual.

Note:

Part I. Configuring GPIO

The PIC16F877A MCU has five (5) I/O port groups namely PORTA, PORTB, PORTC, PORTD and PORTE. All port groups has eight (8) programmable I/O pins except for PORTA and PORTE which has only six (6) and three (3) ports respectively. Some I/O are also multiplexed to other functions as analog input, external interrupt source and serial communication lines (USART and SPI). For more information about I/O ports of the PIC16F877A, go to page 41 of the PIC16F87X data sheet.

The individual I/O ports in a port group is also denoted by the name Rxn where x is the port group letter and n is the bit position. For example, RA0 is an I/O port which belongs in PORTA and it represent the LSB of the port group.

Each I/O port can be programmed as either input or output when used as a general purpose I/O. To program the registers TRISA, TRISB, TRISC, TRISD and TRISE must configured. Each bit in these registers corresponds to the individual I/O ports in the group. Set the bits to the following:

TRISx bit <= ‘0’ - function as an output pin
‘1’ - function as an input pin

For example, if all ports of PORTB will be input, then the value of the latter will be 00000000_2 or 0x00. If RB0 is set to input and the rest are output then the value of TRISB will be 00000001_2 or 0x01.

When using alternate function, the TRISx might be required to be programmed in the specific signal direction required by the function

The following is an example code to program the I/O port using C and writing and reading to/from an I/O port.

```
void main()
{
      unsigned char data;

      OPTION_REG = 0xC0;      // configuring the OPTION register
      TRISA = 0x0F;           // set RA0-RA3 as input, the rest are output
      TRISB = 0x00;           // set all of PORTB as output

      for(;;)

      {

            PORTB = 0x00;     // all ports in PORTB is '0'
            data = PORTA & 0x0F;    // read data in PORTA and mask
            …
            …
            …

      }

}
```

You can also access an individual port and perform write/read operation without having to configure the entire port register. For example, to set only RB0 to '1', you can write `RB0 = 1;` in your code. On the other hand, you can read an individual I/O port for example; `if(!RA0)`, in this case it is evaluating the value of RA0.

Part II. Interfacing Displaytech 204A LCD

(See Lecture Notes for details)

Interface

The Displaytech 204A LCD supports both 8-bit and 4-bit interface. This 8-bit mode requires the entire 8-bit LCD Data Lines <DB7:DB0> to send both instructions and data while the 4-bit mode uses the only the upper bits <DB7:DB4>. The 4-bit mode requires two (2) 4-bit data transfers.

LCD Control Functions

Before configuring the LCD, the necessary control functions shall be written. These are the following:

> **Note:** The functions below are assumed to be for the 8-bit interface.

1. **instCtrl()** - this function will send an 8-bit instruction to the instruction register
2. **dataCtrl()** - this function will send an 8-bit data (character) that is to be displayed on the LCD
3. **initLCD()** - this function will initialize the LCD prior to write

Open Proteus and create a firmware project for PIC16F877A with the filename "LE2-1.pdsprj" and interface the LCD to the microcontroller. Refer to the following connections:

LCD Data Lines (DB7:DB0) -> PORTB (RB7:RB0)
RS -> PORTC (RC0)
E -> PORTC (RC1)

Properly connect the power supply and ground lines of the MCU and LCD.

Open MPLAB IDE and create a new project via the project wizard with the name "CpE 3202-LE2". Create a new source file with the filename "LE2-1.c" and add it to the project. Write the necessary

control functions as stated above. Add the source file to the project. Refer to the schematic diagram in “LE2-1.pdsprj”.

LCD Initialization

In order for the LCD to function properly, it has to be configured specifically as an 8-bit interface. Refer to the **LCD instruction set** and **initialization procedure**. The configuration of the LCD depends on the programmer preference or design requirements. In this exercise, we shall configure the LCD to the following requirements:

- function set: 8-bit interface & dual-line
- entry mode: increment & shift off
- display on: cursor on and blink off

The following is the **initLCD()** function. Supply the necessary instructions to achieve the configuration above. All delays are approximate and can be implemented using a simple loop.

```
initLCD()
{
      delay(1000);                  // LCD startup about 15ms
      instCtrl(__________);         // function set: 8-bit; dual-line
      instCtrl(__________);         // display off
      instCtrl(__________);         // display clear
      instCtrl(__________);         // entry mode: increment; shift off
      instCtrl(__________);         // display on; cursor on; blink off
}
```

LCD initialization shall be done only once upon powering the system. If the LCD failed to initialize, the characters will not be displayed. To test the initialization, run the initialization once as indicated on the main function below.

```
      void main(void)
      {
            …
            …
            initLCD();  // initialize LCD

            while (1)   // endless loop
            {

            }
      }
```

After successfully building the source file “LE2-1.c”, simulate the program in Proteus (“LE2-1.pdsprj”) by loading the generated .hex file to the microcontroller.

If the LCD initialized successfully, the cursor would appear on the first line and first column (due to the display clear instruction). The LCD now is ready for further instruction or displaying characters.

Displaying Characters

The LCD would print a character to the location pointed to by the Address Counter. Each character displayed is stored to the Display Data RAM (DDRAM). The position of the cursor indicates the address of the character to be displayed relative to the display area. Each character position has a permanent address. The cursor can be moved to any position by sending the address of the position desired to the LCD as an instruction. (For more details on LCD character address, see lecture notes).

For example, to display a character on the first line, fifth column on the LCD the instruction “0x84” must be sent to the LCD. This positions the cursor to fifth column of the first line.

To display a character, use the **dataCtrl()** instead. You may use the ASCII code of the character or pass a character directly to the function.

dataCtrl(0x41) or *dataCtrl('A')*

The data sent is basically the address of the characters in the Character Generator ROM (CGROM) which generates the character and is sent to the DDRAM for display. The following snippet will display the string "HELLO!" on the middle of the second line.

```
instCtrl(0xC6);         // move cursor to 2nd line 7th column
dataCtrl('H');          // prints 'H' at current cursor position
                        // then shifts the cursor to the right*
dataCtrl('E');          // prints 'E'
dataCtrl('L');          // prints 'L'
dataCtrl('L');          // prints 'L'
dataCtrl('O');          // prints 'O'
dataCtrl('!');          // prints '!'
```

Write the code above in the "LE2-1.c" source file together with the control functions and the delay(). After successfully building the source code, simulate the program in Proteus ("LE2-1.pdsprj") by loading the generated .hex file to the microcontroller. Observe the output and make sure the correct characters are displayed.

Part III. Interfacing Numeric Keypad Using 74C922

(See Lecture Notes for details)

The 3x4 numeric keypad on the expansion board is connected the MCU via the 74C922 keypad encoder. The encoder automatically scans the each column of the LCD and generate a 4-bit address of the key being pressed. It also has switch debounce function which takes of the switch bounce. A DVABL (Data Available) pin will set to logic-1 if any key is pressed. The OE (Output Enable) will enable the output of the 4-bit data when set to logic-0. Therefore to make sure the data is valid, an inverter will be connected from DVABL to OE in which the 4-bit data is outputted when there is a key press. Refer to the schematic diagram of the expansion board.

Key Addresses

The 74C922 supports up to 4x4 layout or a total of 16 keys. Each of the keys has a 4-bit address. It can also used in 3x4 layout in which keys (address) on the fourth column are inaccessible. The following is address table of the 74C922:

Key	74C922 Data Output (bin)	Hex	Key	74C922 Data Output (bin)	Hex
1	0000b	0x00	7	1000b	0x08
2	0001b	0x01	8	1001b	0x09
3	0010b	0x02	9	1010b	0x0A
A	0011b	0x03	C	1011b	0x0B
4	0100b	0x04	*	1100b	0x0C
5	0101b	0x05	0	1101b	0x0D
6	0110b	0x06	#	1110b	0x0E
B	0111b	0x07	D	1111b	0x0F

Reading Keypad Data

To read the 4-bit keypad data, the DAVBL must be read first. If DAVBL = '1', a key press happens then the 4-bit keypad shall be read and stored on a variable. The keypad data shall be buffered to prevent data loss.

LED+Keypad Exercise

Create a firmware project for PIC16F877A in Proteus with the filename “LE2-2.pdsprj” and interface the numeric keypad to the microcontroller. Refer to the following connections:

Keypad Data	->	PORTD (RD3:RD0)
DAVBL	->	PORTD (RD4)
LEDS (4)	->	PORTA (RA3:RA0)

Make sure to connect an inverter from DAVBL to OE pin of the 74C922. This is to automatically enable the data output of the encoder when any key is pressed.

Create a new source file in MPLAB IDE with the filename “LE2-2.c” and add it to the “CpE 3202-LE2” project. Write a program that will display the key being pressed on the keypad to the LEDs. For example, if key 3 is pressed the LED would display 00011_2. Keys * and # should display 11111_2. Refer to the schematic diagram in “LE2-2.pdsprj”.

After successfully building the source code “LE2-2.c”, simulate the program in Proteus (“LE2-2.pdsprj”) by loading the generated .hex file to the microcontroller. Observe the response of the microcontroller to the key press.

Part IV. Hands-on Exercise

Interface both LCD and keypad (with 74C922) to the PIC16F877A. You decide which port to connect the I/O devices. Write a program that will display the key pressed on the keypad to the LCD. When the LCD is full of characters, it will automatically clear and start displaying characters again.

1. Create a firmware project for PIC16F877A in Proteus with the filename “LE2-3.pdsprj” and construct the required circuit.
2. Draw a flowchart using the flowcharting application of your choice and annotate the necessary information. Export flowchart as PDF as “LE2-3.pdf”.
3. Create a new source file in MPLAB IDE with the filename “LE2-3.c” and add it to “CpE 3202-LE2” project. Write the program in this source file.
4. After successfully building the source code “LE2-3.c”, simulate the program in Proteus (“LE2-3.pdsprj”) by loading the generated .hex file to the microcontroller.
5. Observe the output of the program and make sure it functions as required.
6. Submit the flowchart (“LE2-3.pdf”), Proteus project file (“LE2-3.pdsprj”) and the source code (“LE2-3.c”) in Canvas.

——————————— *End of Practical Activity #2* ———————————————

Assessment

Criteria	Outstanding (4 pts)	Competent (3 pts)	Marginal (2 pts)	Not Acceptable (1 pt)	None (0 pts)
Design	Design of the embedded system was based on a systematic analysis of the problem.	Design of embedded system follows a systematic analysis but some details lacking.	The system was designed with little analysis of the problem.	The design of the embedded was done without proper analysis of the problem.	No DE presented

Functionality	The systems function perfectly.	The system functions with minor issues.	The system has several issues which already affect the functionality.	The presented system is non-functioning.	No DE presented
Firmware	Design of firmware was done systematically using flowcharts, state diagrams etc.	The firmware was designed systematically but missing details in the flowcharts, state diagrams etc.	The firmware design is not supported by flowcharts, state diagrams, no evidence of a systematic process.	There is no evidence of systems analysis and design.	No DE presented

Copyright Information

Author: Van B. Patiluna (vbpatiluna@usc.edu.ph)
Contributors: none
Reviewer(s): Engr. Alvin Joseph S. Macapagal (ajsmacapagal@usc.edu.ph)
Date of Release: February 16, 2021
Version: 1.0.1

References

- Displaytech 204A LCD Data Sheet.
- MM74C922 Data Sheet, Fairchild Semiconductor 1999.
- CpE 3201 Lecture Notes on LCD and Numeric Keypad Interfacing (available on Canvas).

change log:

Date	Version	Author	Changes
September 13, 2018	1.5	Van B. Patiluna	Original laboratory guide for CpE 426N.
February 15, 2021	1.0	Van B. Patiluna	- Revised the activity objectives. - Removed the LKS Expansion Board and other references to MB90F387S MCU. - Changed some of the procedure to reflect the new software tools. - Remove the Mini-Design activity and replaced with Hands-on Exercise.
February 15, 2021	1.0.1	Van B. Patiluna	- Corrected some factual errors. - Added credit to reviewers.