**Laboratory Exercise #6**
Parallel I/O Devices Interfacing

**Exercise Objectives:**

1. Interface parallel I/O devices to the 8086 microprocessor.
2. Implement I/O interfacing using the 8255 Programmable Peripheral Interface IC.
3. Write assembly program to control the parallel I/O devices LCD and keypad.

**Tools Required:**

The following will be provided for the students:
1. PC with Proteus ISIS v8.6 or later installed
2. Programming IDE + 8086 assembler (Emu8086)

**Delivery Process:**

The instructor will conduct a short briefing about the exercise. Knowledge from Unit VII (Interfacing Parallel I/O Devices) will be applied in this exercise, thus the instructor will also provide a short review of the recent chapter.

**Note:**
Some of the contents in this manual are derived with permission from other sources. This document is a guide only. Your answers are to be submitted in a separate document.

# Part I. Theory

LCD Interfacing

*Interface*

The LCD is interfaced to the MCU via a 8-bit interface mode in which Displaytech 204A. This mode requires all 8-bits of the LCD Data Lines (DB0-DB7) to send both instructions and data. 8-bit instruction and data can be sent directly to the LCD in a single cycle.

*LCD Control Functions*

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

1. **INST_CTRL** - this procedure will send an 8-bit instruction to the instruction register
2. **DATA_CTRL** - this procedure will send an 8-bit data that is to be displayed on the LCD
3. **INIT_LCD** - this procedure will initialize the LCD

*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 off and blink off

*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 **DATA_CTRL** instead. You may use the ASCII code of the character or pass a character directly to the function.

Interfacing Numeric Keypad

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 care 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:

Table 1: Key addresses (using 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 buffered to prevent data loss.

# Part II. Interfacing LCD & Keypad

Before starting the activity, make sure you have already completed the circuit used in Laboratory Exercise #5, Activity #3 (“<last name>_LE5-3.pdsprj”).

**Activity #1:**

Instructions:

1. Open the circuit in Laboratory Exercise #5, Activity #3 (“<last name>_LE5-3.pdsprj”). Save the circuit as “<last name>_LE6-1.pdsprj”.

2. Remove the 7-segment displays and connect the LCD to the 8255. Refer to the following:
    - LCD Data (DB0-DB7) - connect to PORTA (PA0-PA7)
    - RS - connect to PORTB (PB0)
    - E - connect to PORTB (PB1)
3. Determine the command byte for the 8255 based on the I/O connections in #2.
4. Write an assembly program to display “HELLO!” In the middle of the second line of the LCD (see Figure 1). Run the **INIT_LCD** function first before using the **INST_CTRL** and **DATA_CTRL** procedures. Save the program as “<last_name>_LE6-1.asm”.


Fig. 1. LCD displaying “HELLO!”.

5. Compile and simulate the program in Proteus and observe the output.

**Activity #2:**

For the this activity, use the design project file in Activity #1 (“<last name>_LE6-1.pdsprj”). Do not proceed if the previous activity is not completed.

Instructions:

1. Open the circuit in Activity #1 (“<last name>_LE6-1.pdsprj”). Do not edit the schematic but save the project as “<last name>_LE6-2.pdsprj”.
2. Connect the keypad with the 74C922* to the 8255. Refer to the following:
    - A,B,C,D - connect to port C of 8255 (PC0-PC3)
    - DAVBL - connect to port C of 8255 (PC4)

    ** follow the proper electrical connections see Appendix*
3. Determine the command byte for the 8255 based on the I/O connections in #2.
4. Save the source code “<last name>_LE6-1.asm” as “<last_name>_LE6-2.asm”
5. Write a main program that will read the keypad data. Then on the middle of the LCD, display the key being pressed. Figure 1 shows ‘8’ being the key being pressed on the keypad. The display will only change when a another key is pressed.

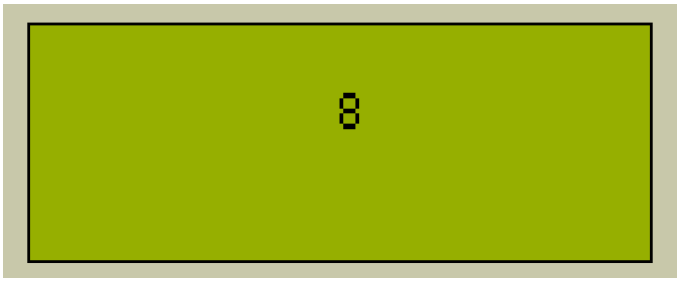



Fig. 2. LCD displaying key ‘8’ being pressed.

6. Compile and simulate the program in Proteus. Observe the output and make sure it functions as required.

**Activity #3**

For the this activity, use the circuit in Activity #2 (<last name>_LE6-2.pdsprj"). Do not proceed if the previous activity is not completed.

Instructions:

1. Open the circuit in Activity #2 ("<last name>_LE6-2.pdsprj"). Do not edit the schematic but save the project as "<last name>_LE6-3.pdsprj".
2. Connect four (4) LEDs to any free I/O port. Although there are free ports in PORTB, it is not recommended since PB0 and PB1 is already used by the LCD control lines RS and E which means that the entire PORTB should be reserved. This is due to the nature of PORTB as an 8-bit output port and it is difficult to set individual bits. Therefore, a second 8255 PPI should be added. Choose an address range where to connect the second 8255 in the decoder. The second 8255 must be programmed as with the first one. Connect the LEDs to the second 8255.
3. Refer to the problem below:

   A computer system with an LCD and numeric keypad controls a soft drink dispenser (via logic signals represented by 4 LEDs). The dispenser has two (2) types of beverage and each has two (2) cup sizes. When the system starts, the choices will be displayed (see Figure 3) then waits for user input.

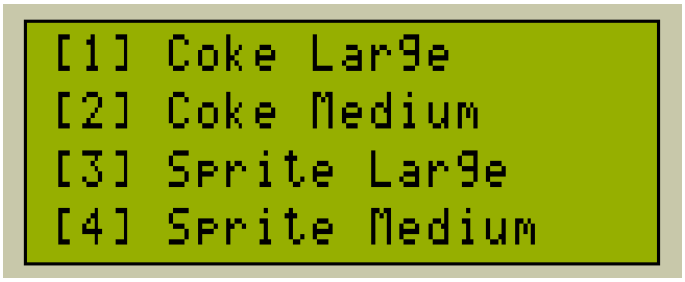



Fig. 3. Main menu.

Table 2 shows the output signals via the LEDs based on the user's choice. The LED has to turn **on** only at a specified duration to simulate dispensing a certain volume of liquid.

Table 2: LED Outputs*

| | Choice | LED1 | LED2 | LED3 | LED4 |
|---|---|---|---|---|---|
| 1 | Coke Large | on (7 s) | off | off | off |
| 2 | Coke Medium | off | on (4 s) | off | off |
| 3 | Sprite Large | off | off | on (7 s) | off |
| 4 | Sprite Medium | off | off | off | on (4 s) |

** In real world application, the signal will drive a solenoid valve to dispense the beverage.*

After the user inputs the choice (1-4), the corresponding LED will turn for the given duration then the LCD will display a status as shown in Figure 4. After the time duration, the LED turns of and the display will go back to the beverage menu (see Figure 3) and waits for user input.

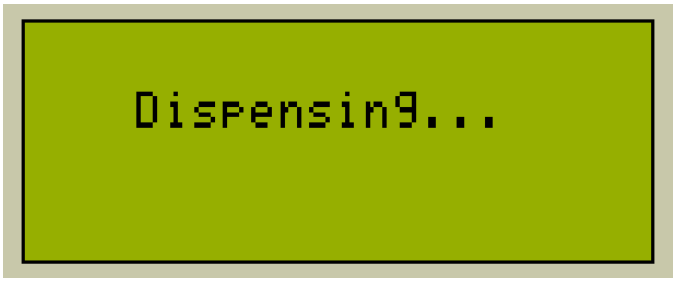



Fig. 4. Display while dispensing.

4. Write an assembly program to perform the functions as given in #2. Save the program as “<last name>_LE6-3.asm”.
5. You can use a simple delay using loops for the LED “on” duration. Make sure that the timing is close to the actual.
6. [OPTIONAL] You can also use the 8253 PIT for a very accurate timing. It can be also used for the delay used in the LCD functions. Use a 4KHz clock input to the 8253.
7. [OPTIONAL] While the system is dispensing, you can also display a count down in seconds just below the “Dispensing…” string (see Figure 5).


Fig. 5. Count down time display.

**Submission Instructions:**

- Accomplish the laboratory report. Include the solutions, data, source codes and answers to questions. Save the report as PDF with the filename “<last name>_LE6.pdf”.
- Submit the following in Canvas in the following order:
    a. <last name>_LE6.pdf
    b. <last name>_LE6-1.asm
    c. <last name>_LE6-1.pdsprj
    d. <last name>_LE6-2.asm
    e. <last name>_LE6-2.pdsprj
    f. <last name>_LE6-3.asm
    g. <last name>_LE6-3.pdsprj

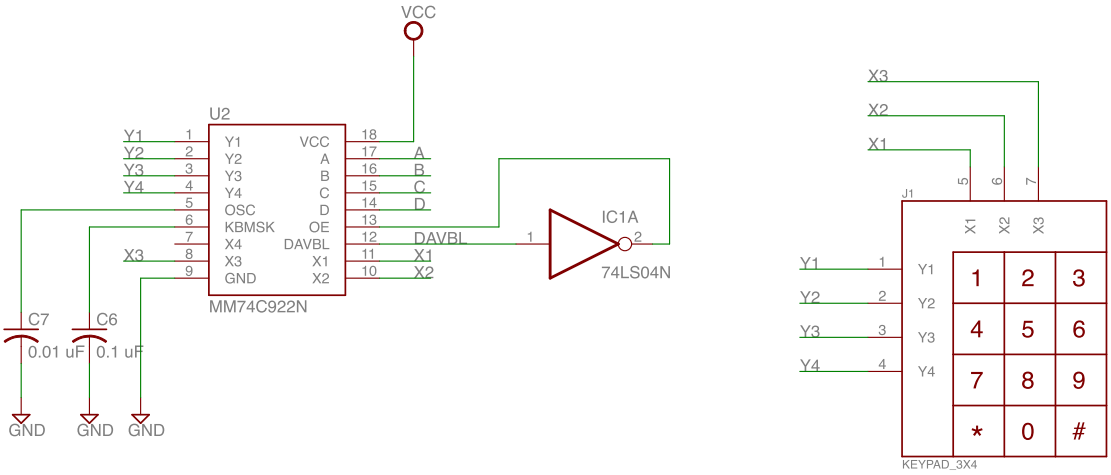
# Assessment

| Criteria | 1.0<br>**Outstanding** | 2.0<br>**Competent** | 3.0<br>**Marginal** | 4.0<br>**Not Acceptable** |
|---|---|---|---|---|
| Circuit Integrity | The circuit was properly constructed. | - | Circuit has issues in terms of construction. | Circuit was constructed poorly. |
| Circuit Simulation | Simulation results are accurate with no issues. | Simulation results are partially correct. | - | Simulation results are inaccurate. |
| Correctness of Program | The program is correct with no issues. | The program has minor issues. | The program has several issues but manage to achieve some of the functions required. | Program did not work as intended in the exercise. |

*End of Laboratory Exercise #6*

# Appendix

**Electrical Connection of 74C922**
(with 3x4 keypad)




# Copyright Information

*Author: Van B. Patiluna (vbpatiluna@usc.edu.ph)*
*Contributors: none*
*Date of Release: November 3, 2021*
*Version: 1.2.2*

# References

- *Displaytech 204A LCD Datasheet*

Change log:

| Date | Version | Author | Changes |
|---|---|---|---|
| August 3, 2020 | 1.0 | Van B. Patiluna | - Initial draft. |
| October 29, 2020 | 1.1 | Van B. Patiluna | - Revisions based on the major changes in Laboratory Exercise #5. |
| November 9, 2020 | 1.1.1 | Van B. Patiluna | - Revised the I/O connections to the 8255 and rename the LCD control procedures.<br>- Added Appendix |
| August 27, 2021 | 1.1.2 | Van B. Patiluna | - Corrected grammatical and typographical errors. |
| October 21, 2021 | 1.2.0 | Van B. Patiluna | - Revised Activity #3<br>- Replaced Figures 1 & 2. |
| November 1, 2021 | 1.2.1 | Van B. Patiluna | - Revised item 2 in Activity #3.<br>- Added optional items in Activity #3. |
| November 3, 2021 | 1.2.2 | Van B. Patiluna | - Corrected the LCD intialization requirement from 4-bit to 8-bit interface, cursor will be off. |