Homework-1
"Guess the magic number!"

Revision 1.0

Due: Sunday, Mar 30, 2014, 23:55:50hrs
Submission: ***via COW***

The purpose of this assignment is to familiarize you with **basic input/output operations** by implementing a simple single player game.

Any clarifications and revisions to the assignment will be posted to the *metu.ceng.course.336* newsgroup.

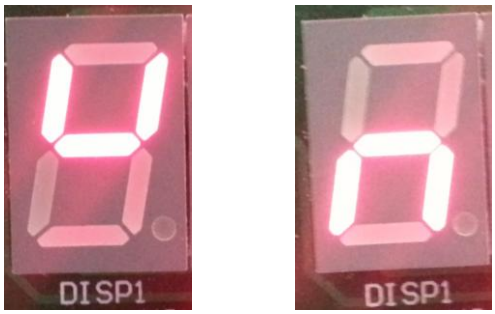
Your Mission

Your mission is to implement a game in which the player is expected to guess a number that is randomly generated by the PIC. Detailed specifications of the game are as follows:

- The 7-Segment displays DISP1, DISP2, DISP3 and 6 LEDs should be used as output devices while SW00, SW04, SW05, SW06, SW07 and SW11 buttons used as input devices.
- When the PIC is powered up, DISP1, DISP2 and DISP3 should show "-" and all LEDs should be turned off until SW00 is pressed. After you press SW00, PIC will generate a random number between 0-99, after which the game will start.
- When the game is started by pressing SW00, 6 LEDs should be turned on (from LED0 to LED5), indicating 6 units of initial energy. DISP1 should keep the character "-", and on DISP2 and DISP3 number "0" should appear.
- After this step, the player is expected to find the randomly generated number by entering his/her own guesses. Number entry will be performed by showing the number using DISP2 and DISP3 for a two digit decimal number. DISP2 and DISP3 will represent the tens and units digits of the number, respectively. For the numbers between 0 and 9, the notation will be as "01, 02 … 09". You will use SW04 to increment the number on DISP2 and SW05 to decrement it. Similarly, SW06 will be used to increment the number on DISP3 and SW07 to decrement it. After you see the desired number on displays (e.g. for "23" you should see "2" on DISP2 and "3" and DISP3), then press SW11 to enter it. DISP1 should continue to show "-" during this period.
- When a number is entered, if it is correct, DISP2 and DISP3 should flash this number approximately for **three** seconds. By flashing, we mean that the number should appear on DISP2 and DISP3 and then disappear repeatedly, for a total of approximately three seconds. The appear/disappear period is up to you, but after a total of three seconds flashing, the 7-Segment displays should show "End", with the LEDs showing the remaining energy level.
- If the player enters an incorrect number, s/he loses **one** unit of energy. So, you should turn off the left most led (starting from LED5). If the energy of the player becomes 0 (after 6 incorrect attempt), then s/he loses the game. After a loss, the 7-Segment displays should show "End", with all LEDs turned off.
- After an incorrect entry, if the player still has some energy, a **hint** should appear on DISP1 that leads the player to the correct number. This hint tells the player that if his/her guess is below or above the randomly generated number. If the guess is smaller than the randomly generated number, DISP1 shows the character in **Figure 1(a)** (It means that the next guess should be greater than the previous one). If the guess is greater than the randomly

generated number, DISP1 shows the character in **Figure 1(b)** (It means that the next guess should be smaller than the previous one). DISP2 and DISP3 should keep the lastly tried number. Then the player enters the next number by modifying the last number on DISP2 and DISP3 according to the hint. When the player hits a button to increment or decrement a digit, the hint should disappear and DISP1 should show the character "-".

- To start a new game, the player can use the reset button on the Board.



(a) (b)

Figure 1. Hints to help the player to enhance its guess. (a) and (b) mean that the next guess should be greater and smaller than the previous one, respectively.

Coding Rules

- You will code your program using PIC assembly language.
- Your program should be written for PIC16F877 working at 20 MHz
- You are provided with a template .asm file named "hw1_##.asm" (## represents your group number). You should add this .asm file as a source file in your MPLAB X IDE project. You are expected to write your code in this file. Moreover, you will use an additional header file (randomGenerator.INC) to generate a number randomly. You should add this .inc file as a header file in your MPLAB X IDE project. "hw1_##.asm" contains required function calls, some initial configurations and some directives. (You can download these files from Homework 1 page on the COW).
- Timer0 interrupt is used in the template .asm file to generate a random number, however the use of interrupts other than this library provided interrupt is prohibited in this homework. You have to use polling to control buttons. For delays, you have to calculate the amount of cycles in a code segment (like the delays in recitation sample codes) therefore the amount time spent (See the Hint in Grading section for how to calculate the time spent by a code segment in simulator.).
- You should be aware of whether the player pushes and releases the SW00 button. If so, you should call the subroutine "assignRandomNumber" as shown in "hw1_##.asm". Then the random number will be saved in register "023h" called "randNumber".

Resources

- PIC 16F877 Datasheet
- PIC Development Tool User Manual
- PIC Development Tool Programming Manual
- Course web page and newsgroup

Mini How-to

Push Buttons

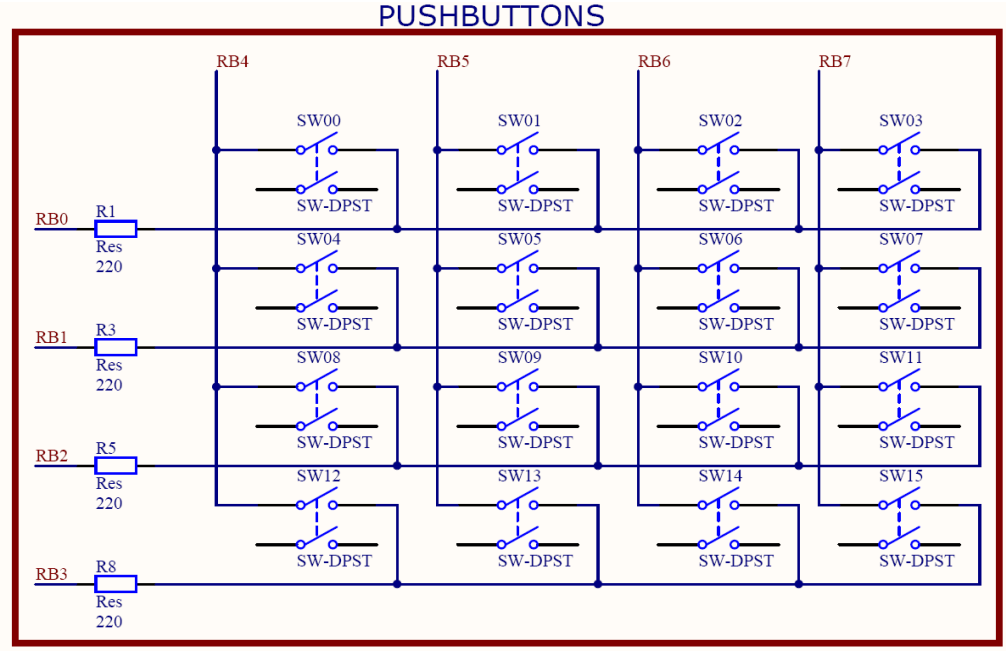


Figure 2. Schematic diagram of push buttons on the development board. Taken from User Manual of the development board which is available at COW.

Development board has 16 push buttons whose schematic diagram is given in Figure 2 (Refer to the complete schematic diagram of the development board in the User Manual whenever you need during whole semester.). In order to use push buttons, you should use PORTB. As you know, each of the PORTB has a weak internal pull-up. So, you must turn on all the pull-ups by clearing RBPU bit (OPTION_REG<7>). Pull-ups are automatically turned off when the port pin is configured as an output. Note that the pull-ups are disabled on a hard reset or start-up.

If you want to control whether SW00 has pressed or not (Do not forget that you will control SW00, SW04, SW05, SW06, SW07 and SW11 in this homework!), one possible approach is as follows:

- Configure RB4 as input and RB0 as output, and turn on pull-ups.
- Clear RB0 and check RB4. If RB4 becomes 0, it means that SW00 has pressed.

As indicated in Recitation-3, please note that the buttons are mechanical components causing some noisy signals during press and release moments, and PORTB pins do not have Schmitt Trigger input buffers to handle such noisy signals. You should do some extra tricks to properly handle the buttons.

Note also that since some of the pins connected to pushbuttons are shared with the microcontroller of USB programmer on the development board, you should run your program after connecting your development board to UsbPicProg software via clicking `Action->Connect` so that the microcontroller of USB programmer leaves the control of these shared pins to you.

In your development boards RB4 pin is connected to RC2 pin. Therefore you have to configure RB4 and RC2 properly for correct operation of buttons.

7-Segment Displays and LEDs

There are three common cathode 7-Segment displays mounted on the development board. PORTD and PORTE are used as data bus and selection pins, respectively, as illustrated in Figure 3. Notice that PORTD pins are connected to all displays.

As an example, if you want to show number "3" on rightmost display (DISP3), you should first select it by setting RE0 and clearing RE1 and RE2, then send binary "01111001" (0x79) to PORTD. Hence, f, e and dp (dot point) segments on DISP3 will be turned off, and a, b, c, d and g segments will be turned on. Note that RD7 pin is used for dp of displays. Also note that PORTE is configured as analog input on a hard reset or start-up. You should inspect ADCON1 register and configure it properly.

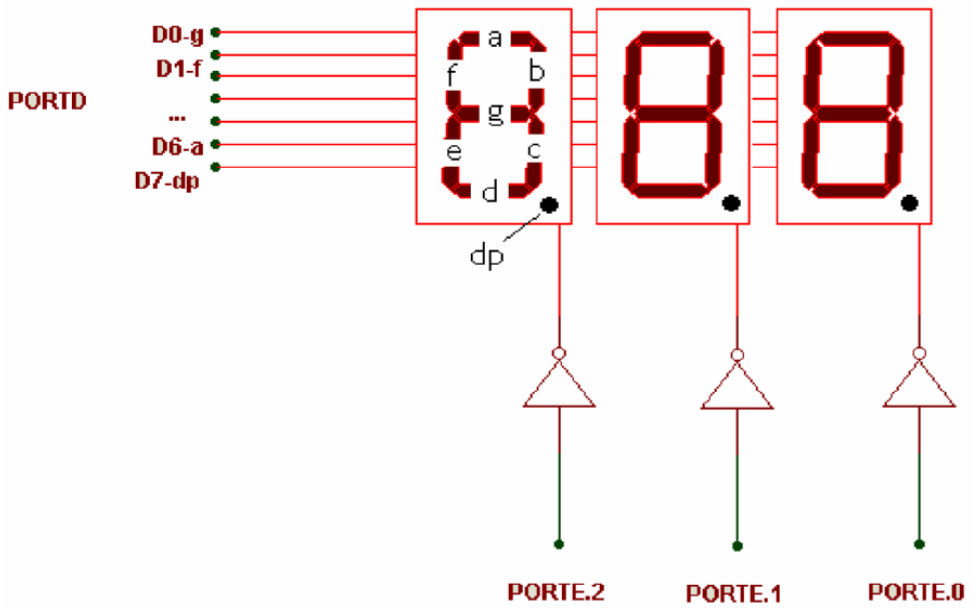


Figure 3. Connections for 7-Segment displays in the development board.

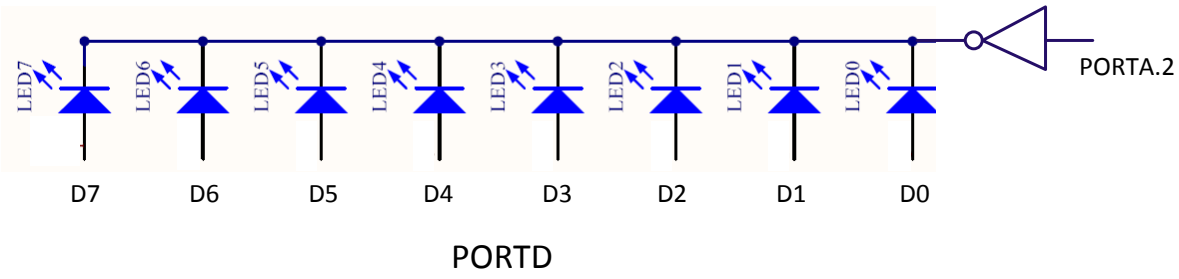


Figure 4. Connections for 8-LED array in the development board.

There are 8 LEDs on the development board whose selection pin is RA2 as illustrated in Figure 4. PORTD is used as the data bus similar to 7-Segment displays. To be able to turn on the LEDs you should configure RA2 as digital output and set it.

If you want to show, for instance some value on LEDs and `123` on the displays, you should first select only LEDs and write the desired value to PORTD, and wait for a while. After that you should select only DISP1, write the byte necessary to turn on segments for `1`to PORTD, and again wait for a while. Then you should do the same for `2` and `3` on DISP2 and DISP3, respectively. This is illustrated in Figure 5. If you adjust "on" times properly and repeat on-off cycles continuously, you show your value on LEDs and `123` on the displays in a smooth manner without flicker.

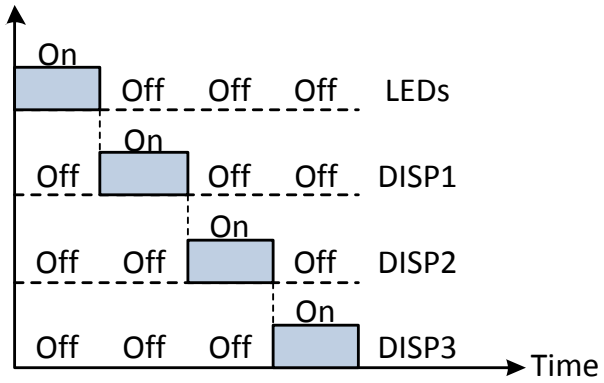


Figure 5. Graphical illustration for showing characters on LEDs and 7-Segment displays simultaneously.

Hand In Instructions

- You should submit your code as a single file named as `hw1_##.asm` through COW where ## represents your group number. Do not forget replace ## with your group number.
- At the top of `hw1_##.asm`, you should write ID, name and surname of **both group members and group number** as commented out.

Only one of the group members should submit the code. Please pay attention to this, since if both members make submission, speed of grading process will be negatively affected.

Grading

Total of the homework is 100 points. For grading we will compile and load your program to the development board and run it for the following checks.

- 20 points for proper displaying the characters on 7-segment display. Too low/high brightness and flicker will reduce your points.
- 25 points for proper control of push buttons. Only SW00, SW04, SW05, SW06, SW07 SW11 will be pressed during grading. You do not have to manage other buttons.
- 15 points for energy on the LEDs.
- 40 points for functional correctness of the game and its ease of use.

Group members may get **different** grades. We reserve the right to evaluate some or all of the groups to determine the contribution of each group member to the assignment.

Hints

- You can use “Stopwatch” tool of the simulator in MPLAB X IDE to measure the time spent by a code segment as explained in Recitation-3. You can reach this tool from `"Windows -> Debugging -> Stopwatch"` menu. But, before starting simulator you have to configure your clock speed to 5 MHz from `"Project properties -> Simulator -> Processor Frequency"` (Since 20 MHz main clock frequency is divided by 4 inside microcontroller and one instruction cycle is equal to 4 cycle of 20 MHz clock signal, you are setting “Processor Frequency” to 5 MHz). You can reach the project properties by right clicking to the project name in the left side “Projects” panel and selecting properties. By putting breakpoints on the lines between which you want to measure the amount of time spent by that code segment and running the code between these two breakpoints, you can see the time spent in stopwatch window.
- You can access the MPLAB X IDE and UsbPicProg on inek machines. You can open the MPLAB X IDE and UsbPicProg by typing "mplab_ide" and "usbpicprog"on command line.

Cheating

We have zero tolerance policy for cheating. People involved in cheating will be punished according to the university regulations.

Cheating Policy: Students/Groups may discuss the concepts among themselves or with the instructor or the assistants. However, when it comes to doing the actual work, it must be done by the student/group alone. As soon as you start to write your solution or type it, you should work alone. In other words, if you are copying text directly from someone else - whether copying files or typing from someone else's notes or typing while they dictate - then you are cheating (committing plagiarism, to be more exact). This is true regardless of whether the source is a classmate, a former student, a website, a program listing found in the trash, or whatever. Furthermore, plagiarism even on a small part of the program is cheating. Also, starting out with code that you did not write, and modifying it to look like your own is cheating. Aiding someone else's cheating also constitutes cheating. Leaving your program in plain sight or leaving a computer without logging out, thereby leaving your programs open to copying, may constitute cheating depending upon the circumstances. Consequently, you should always take care to prevent others from copying your programs, as it certainly leaves you open to accusations of cheating. We have automated tools to determine cheating. Both parties involved in cheating will be subject to disciplinary action. [Adapted from http://www.seas.upenn.edu/~cis330/main.html]