Middle East Technical University | Department of Computer Engineering

CENG 336

Int. to Embedded Systems

March 2020

Take Home Exam 2

Revision 1.0

Due date: 5 May 2020, 23:55
Submission: **via ODTUCLASS**

1 Updates

Due to the COVID-19 pandemic, you will develop THE2 on MPLAB X IDE simulation environment. The updated sections are red colored to be easily noticed.

- The game area is now smaller (see 4.1). **Note that, the given figures and the rules in text is based on the full game area. However, you should apply the same game rules on the mini game area.**
- The number of balls to be created has been reduced (see 4.3).
- Interrupt intervals are tolerated up to 100 ms (see 4.3).
- You will not be able to simulate 7-segment display, BUT you still expected to update the data on PORTJ and PORTH.
- Please avoid implementing large code blocks in your ISRs.
- There will NOT be any additional recitation related to this HW. You could use your lectures notes from recitation1 and recitation2. You could also benefit from your experience from HW1.

2 Objectives

This assignment concerns basic input/output operations together with programming with **interrupts and timers**. This will done in the context of implementing a simple *catching the ball* game (see Figure 1). Clarifications and revisions on the content, scope and requirements of the assignment will be posted to the course page discussion forum in ODTUCLASS.

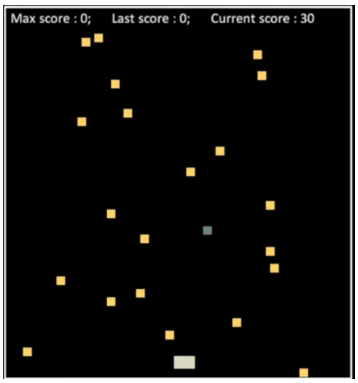

Figure 1: A screenshot from *catching the ball* game.

3 Scenario

In the context of the game scenario, one player directs a bar at the bottom of a predefined game field to catch falling balls. Balls will be created continuously and fall from top to bottom. The player should move the bar **only horizontally** to catch balls onto the bar. The game consists of **3 levels**. The player will start with 5 health points (HP). For every missed ball the player will lose one HP. You will display the current game level and HP of the player on 7-segment display screen. **Once, the player misses 5 balls (or no HP left), or survives against all the created balls game will be over**. You will use buttons to move the bar. In order to create and move the balls you will use the **Timer0 interrupt**.

4 Specifications

4.1 Initial Configuration

The game will be played in the area of the LEDs trough RA0-RA5, RB0-RB5, RC0-RC5, RD0-RD5. Figure 2 illustrates the border of the game field where the bar and the balls will move across these LEDs. You could see the initial configuration of the game in Figure 3. Board should start with this configuration. As it could be seen from the figure, the bar is represented with two consecutive LEDs and placed on RA5-RB5. Also the game level and HP of the player are set 1 and 5 respectively at the beginning. The first digit (D3) of the 7-segment display shows the game level. The last digit (D0) of the 7-segment display shows the remaining HP of the player.**The game should start as soon as RG0 button is pressed.** An example flow of the game is illustrated in Figures through 3 to 9.

4.2 Moving the Bar

You should use following button configurations to move the bar. Each time when the one of the given buttons pressed, the bar should move according to the button specification. **The bar will move horizontally between RA5 and RF5**. You should not cross the borders of the game field.

- You should use RG2 button to move the bar right. Let's assume that the bar is on RD5-RE5 and RG2 button is pressed, after that the bar should be on RE5-RF5.
- You should use RG3 button to move the bar left. Let's assume that the bar is on RD5-RE5 and RG3 button is pressed, after that the bar should be on RC5-RD5.

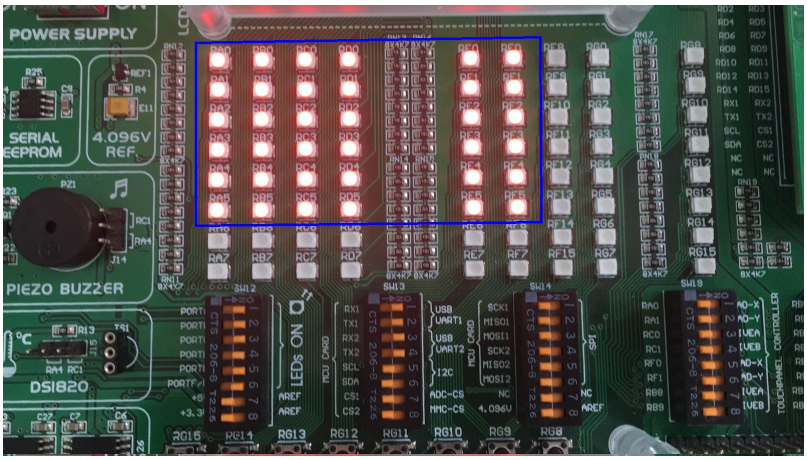

Figure 2: Predefined game field.

4.3 Creating and Moving the Balls

- Each ball should be created on one of the RA0, RB0, RC0, RD0, RE0 and RF0 LEDs.
- The balls should move **vertically**. Let's assume that we created a ball on RD0. The next location of this ball is RD1. Full path of this ball is RD0-RD1-RD2-RD3-RD4-RD5. After RD5, the ball should be disappeared.

  The ball should move from its current location to the next location according to the following rules:

  - **For Level-I**:
    - You should create a new ball in every 500 ms.
    - You should move the balls to the next location in every 500 (+/−100) ms.
    - You should create 5 balls at this level.
  - After the first 10 balls, the second level starts. **For Level-II**:
    - You should update the 7-segment display for game level.
    - You should create a new ball in every 400 ms.
    - You should move all the balls to the next location in every 400 (+/−100) ms.
    - You should create 10 balls at this level.
  - After the first 25 balls, the third level starts. **For Level-III**:
    - You should update the 7-segment display for game level.
    - You should create a new ball in every 350 ms.
    - You should move all the balls to the next location in every 350 (+/−100) ms.
    - You should create 15 balls at this level.
- **You should use the Timer0 interrupt to create these 500ms, 400ms, 350ms time intervals/durations. Please see the Timer0 configuration rules in Section 4.**

4.4 How to Create Balls Randomly?

For the sake of an appealing game, we need to create balls randomly on one of the RA0, RB0, RC0, RD0, RE0 and RF0 LEDs. For this purpose, you will use the value of Timer1 interrupt at the time of RG0 button is pressed at the beginning of the game. **You should implement Timer1 as 16 bit**. Following is the algorithm. For the simplicity the algorithm is explained with 8-bits.

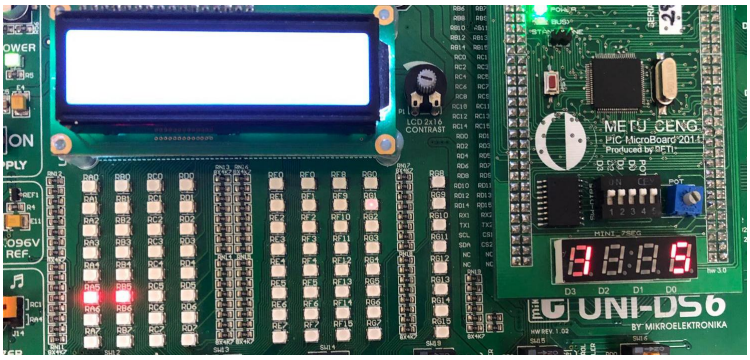

Figure 3: Initial configuration. The bar should be on RA5-RB5 at the start. The first digit (D3) of the 7-segment display shows the game level and the last digit (D0) of the 7-segment display shows the remaining HP of the player.

1. At the beginning of the game, read the value of Timer1 once RG0 button is pressed and save its value. Let's assume that this value is 10001101.
2. Take the rightmost 3-bit (i.e. 101).
3. Apply modulo operation to the rightmost 3-bit. Resulting value indicates where to create a new random ball on the first row of the game field (i.e. (101 mod 6) = 5 → create the ball on RF0).
4. Shift right the value by N. N is 1, 3 and 5 for the Level-I, Level-II and Level-III respectively.
   - if it is Level-I, N is 1, and the new value after the shift operation is 11000110.
   - if it is Level-II, N is 3, and the new value after the shift operation is 10110001.
   - if it is Level-III, N is 5, and the new value after the shift operation is 01101100.
5. Every time when you need to create a new random ball repeat the steps **2**, **3** and **4**.

4.5 Updating the Scores

The player will lose one HP for every missed ball. You should update the 7-segment display immediately after each missed ball. 7-segment display should show the game level through the game. You should update the 7-segment display as soon as the game level changed.

4.6 What Happens when the Game is Over?

Once the game is over go to the initial configuration as in Figure 3 and wait the player to press on RG0 to start a new game.

4.7 Switch Configuration

You should use the switch configuration given in Figure 10.

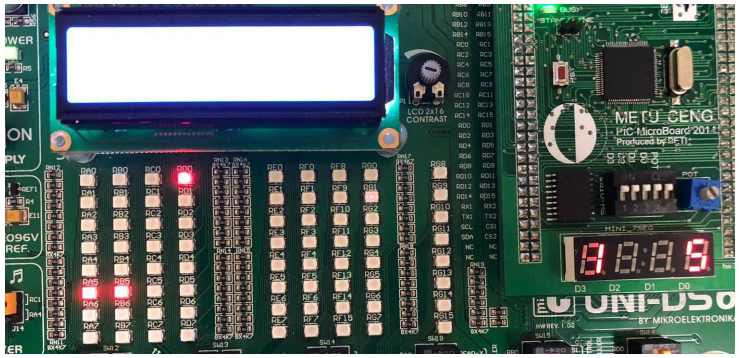

Figure 4: A random ball is created on **RD0**.

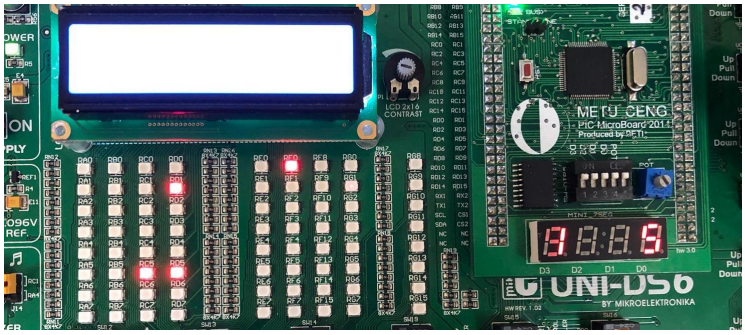

Figure 5: Another random ball is created on RF0. The first ball moved on. The player moved the bar to the right.

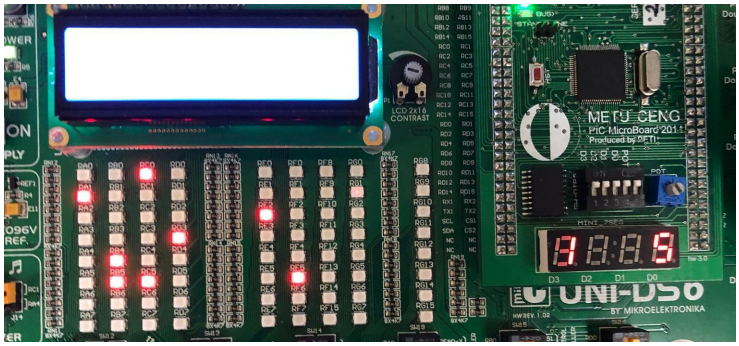

Figure 6: The player misses the ball on RF5.

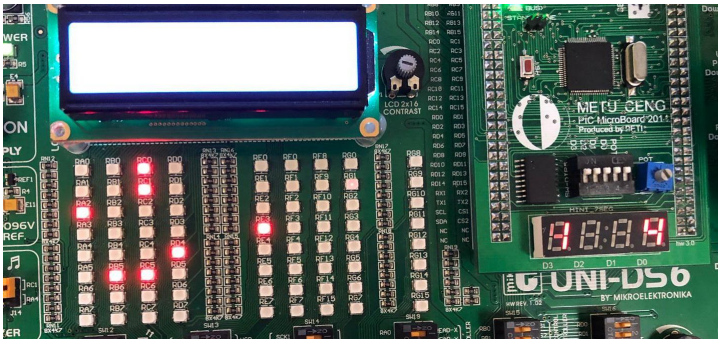

Figure 7: Player lost one HP and the remaining HP is updated on 7-segment display. A new random ball is created on RC0. The other balls moved on.

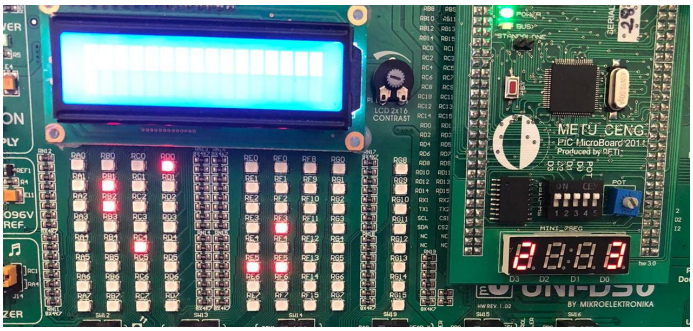

Figure 8: A sample screenshot from Level-II.

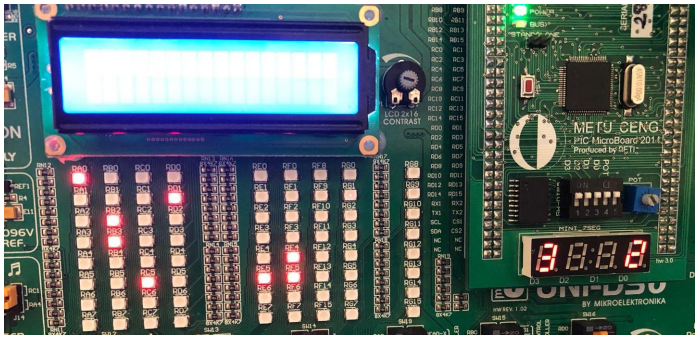

Figure 9: A sample screenshot from Level-III.

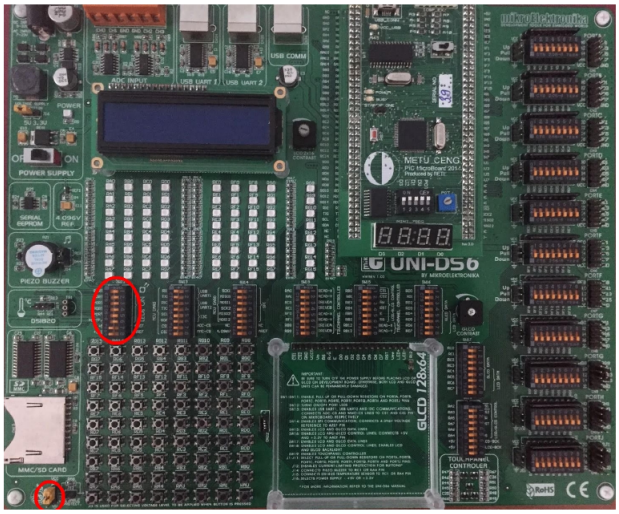

Figure 10: Switch Configuration.

5 Implementation and Regulations

- You will code your program using PIC assembly language.
- Your program should be written for **PIC18F8722** working at **40 MHz**.
- You will use nine ports in this assignment; PORTA, PORTB, PORTC, PORTD, PORTE, PORTF, PORTH, PORTJ and PORTG.
- PORTA, PORTB, PORTC, PORTD, PORTE and PORTF will be used to show the bar.
- PORTA, PORTB, PORTC, PORTD, PORTE and PORTF will be used to show the balls.
- PORTH and PORTJ will be used as output ports to show numbers on 7-Segment displays. You can find some useful information on **Hints** section.
- PORTG will be used to move the bar.
- You are not expected to use interrupts for the buttons (PORTG). You could have button tasks based on states.
- You should use the Timer0 interrupt as stated above and configure **the Timer0 to work in 8-bit mode**.
- You should use the Timer1 interrupt as stated above and configure **the Timer1 to work in 16-bit mode**.
- It is beneficial to avoid function calls in Timer ISRs. You may use some counters or flags and do the function calls or calculations in your main routine. For example, if you call a display subroutine in ISR, measuring the correct time interval in ISR will become harder.
- You will get most of your points from the proper usage of **Timer0 and Timer1** together with the performance of buttons.
- When you are writing your code, you can use the lecture notes on Input/Output and Interrupts, Recitation documents. It is also highly recommended that you make extensive use of the PIC data sheet, MCDEV Kit User Manual and Programming Manual. Please consult these resources first before you ask any questions to the assistants or on the forums.

6 Hints

6.1 7-Segment Displays and LEDs

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

As an example, if you want to show number “4“ on leftmost display (D3), you should first select it by setting RH0 and clearing RH1, RH2 and RH3, then send binary ”01100110” to PORTJ. Hence, a, d, e segments on D3 will be turned off, and b, c, f, g segments will be turned on. Note that RJ7 pin is used for dp of displays. Also note that there is no dp on D3 (leftmost 7-segment display) and there are 3 dp on D1 which run simultaneously.

If you want to show, for instance some value(1234) on the displays, you should select only D0 by using PORTH, write the byte necessary to turn on segments to PORTJ, and wait for a while. Then you should do the same for D1, D2 and D3. This is illustrated in figure below. If you adjust ”on” times properly and repeat on-off cycles continuously, you show your values on the displays in a smooth manner without flicker.

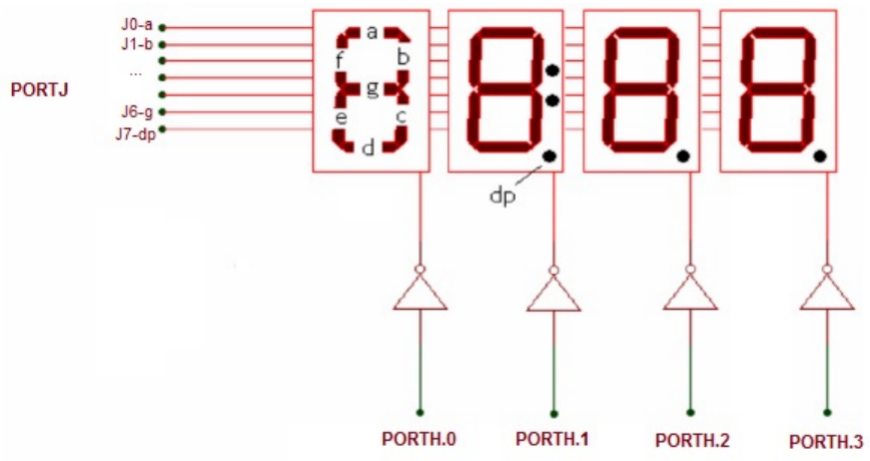

Figure 11: Connections for the 7-Segment displays in the MCDEV board.

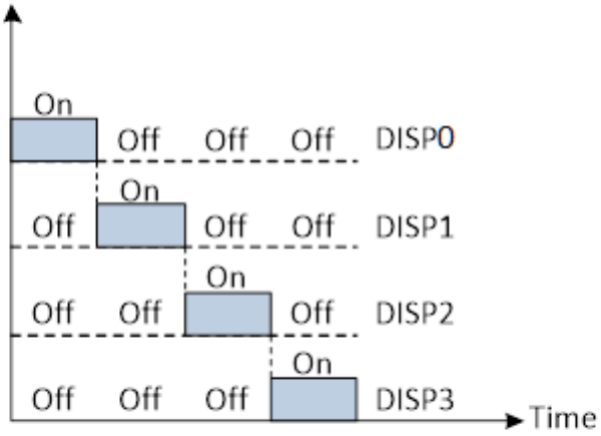

Figure 12: Graphical illustration to show characters 7-Segment displays simultaneously.

7 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]