Middle East Technical University | Department of Computer Engineering

CENG 336

Int. to Embedded Systems

March 2019

Take Home Exam 2

Revision 1.0

Due date: 12 April 2019, 23:55
Submission: **via ODTUCLASS**

1 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 version of Space Impact 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.

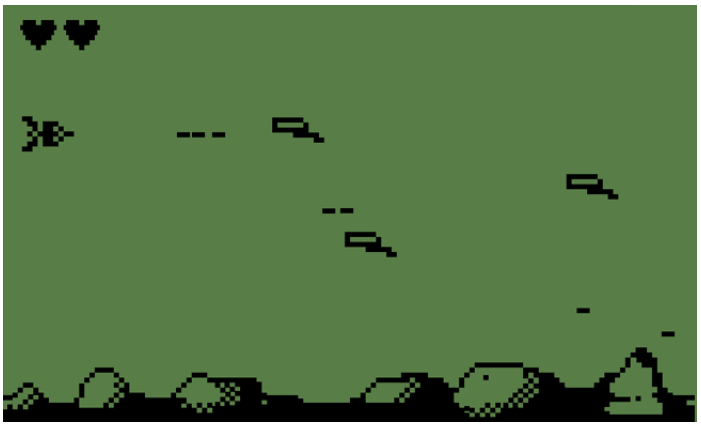

Figure 1: A screenshot from Space Impact game.

2 Scenario

In the context of the game scenario, the player directs his/her spaceship in order to avoid from coming asteroids. The spaceship could move **only vertically** and it could fire the laser beams to destroy the asteroids. For every destroyed asteroid the player will get a score point. You will display score for the player on 7 segment display screen. **Once, an asteroid hits the spaceship, the game will be over**. You will use buttons to fire the laser beams and move the spaceship. In order to create and move asteroids you will use the **Timer0 interrupt**.

3 Specifications

3.1 Initial Configuration

The game will be played in the area of the LEDs trough RA0-RA5, RB0-RB5, RC0-RC5, RD0-RD5, RF0-RF5. Figure 2 illustrates the border of the game field where the asteroids 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 spaceship is represented with a single LED. Also the score for the player is set zero initially. **The game should start as soon as RG0 button is pressed.**
An example flow of the game is illustrated in Figures through 3 to 8

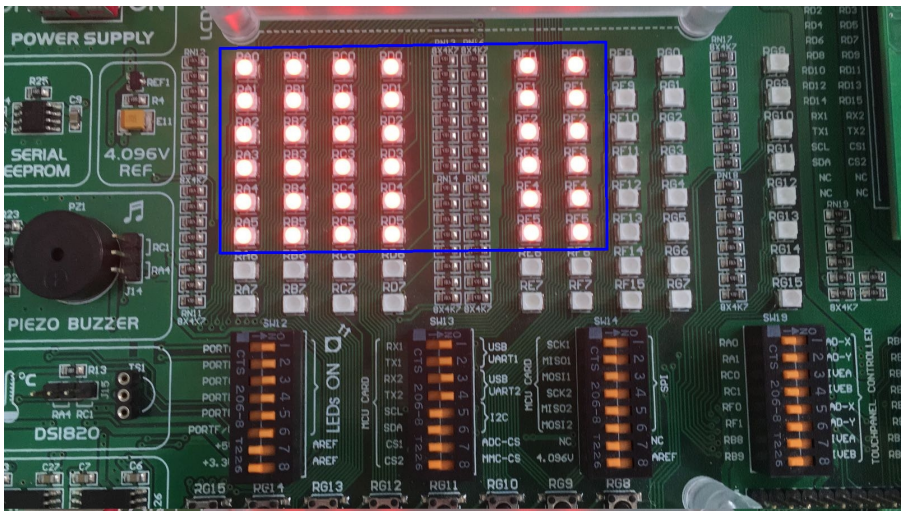

Figure 2: Predefined game field.

3.2 Moving the Spaceship

You should use following button configuration to move the spaceship. Each time when the one of the given buttons pressed, the spaceship should move according to the button specification. **The spaceship will move vertically between RA0 and RA5**. You should not cross the borders of the game field.

- You should use RG3 button to move the spaceship up. Lets assume that the spaceship is on RA2 and RG3 button is pressed, after that the spaceship should be on RA1.
- You should use RG2 button to move the spaceship down. Lets assume that the spaceship is on RA2 and RG2 button is pressed, after that the spaceship should be on RA3.

3.3 Creating and Moving the Asteroids

- Each asteroid should be created on one of the RF0, RF1, RF2, RF3, RF4 and RF5 LEDs.
- The asteroids should move **horizontally**. Lets assume that we created an asteroid on RF3. The next location for this asteroid should be RE3. **Full path of this asteroid should be RF3-RE3-RD3-RC3-RB3-RA3.** After RA3, the asteroid should be disappeared.
- The asteroids should move from its current location to the next location according to the following rules:

- The first 10 asteroids should move to the next location in every **500 ms**.
- After the first 10 asteroids, newly created 20 asteroids should move in every **400 ms**.
- After the first 30 asteroids, newly created 20 asteroids should move in every **300 ms**.
- After the first 50 asteroids, the rest of the newly created asteroids should move in every **200 ms**.
- **You should use the Timer0 interrupt to measure these expected durations.**

3.3.1 How to Create Asteroids Randomly?

For the sake of an appealing game, we need to create asteroids randomly on one of RF0, RF1, RF2, RF3, RF4 and RF5 LEDs. For this purpose, you should 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 4-bit.

1. Read the value of Timer1 once RG0 button is pressed and save its value. Lets assume that this value is 1011.
   (a) Take the rightmost 3-bit (i.e. 011).
   (b) Apply modulo operation to the rightmost 3-bit. Resulting value indicates where to create a new random asteroid on RF (i.e. 011 % 6 = 3 –> create the asteroid on RF3).
2. Shift right the value by one (now it is 1101) and repeat the steps 1a and 1b.
3. After each 10 shift operation, complement the value. Repeat the steps through 1a, 1b and 2.

In order to calculate modulos you need to implement a function. As an alternative to the modulo operation you could use a look-up table.

3.4 Firing the Laser Beams

- **You will use RG1 button to fire laser beams**.
- A laser beam is represented by a single LED.
- The laser beam should move horizontally at the same speed of the asteroids.
- Lets assume that the spaceship is on RA3 and it fires a laser beam when RG1 button is pressed. In this case the next location of the laser beam will be RB3.
- Again lets assume that, this laser beam hits an asteroid at RD3, then both asteroid and laser beam should be disappeared. Note that, in this case the player will get a score point.

3.5 Updating the Scores

The player will have a score for every destroyed asteroid by firing laser beams. **You should update 7 segment display after each score**.

3.6 What Happens when the Game is Over?

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

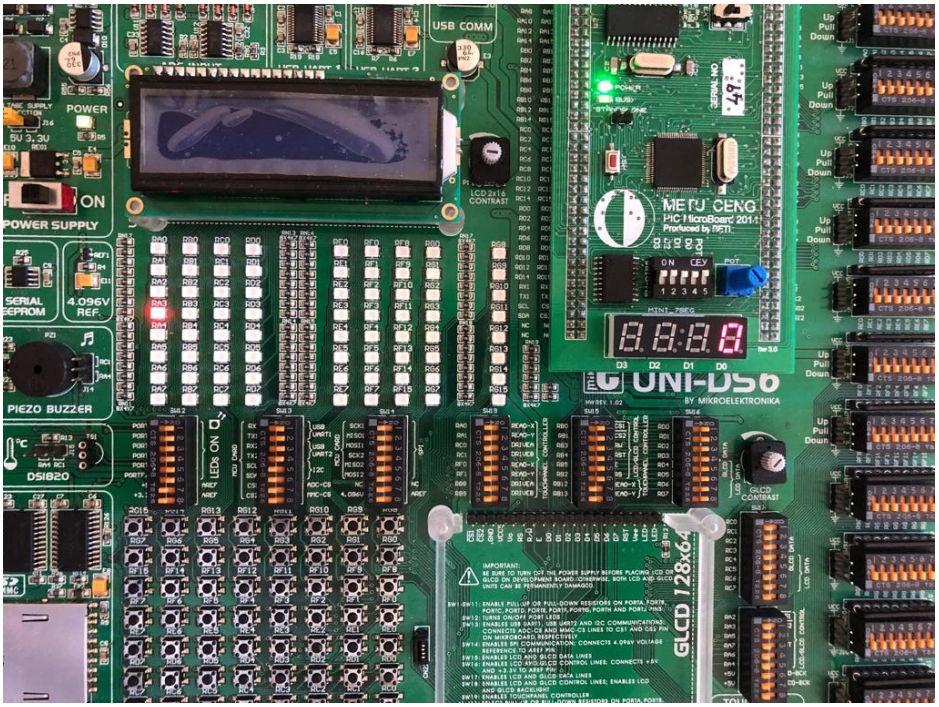

Figure 3: Initial configuration.The spaceship should be on **RA3** at the start.

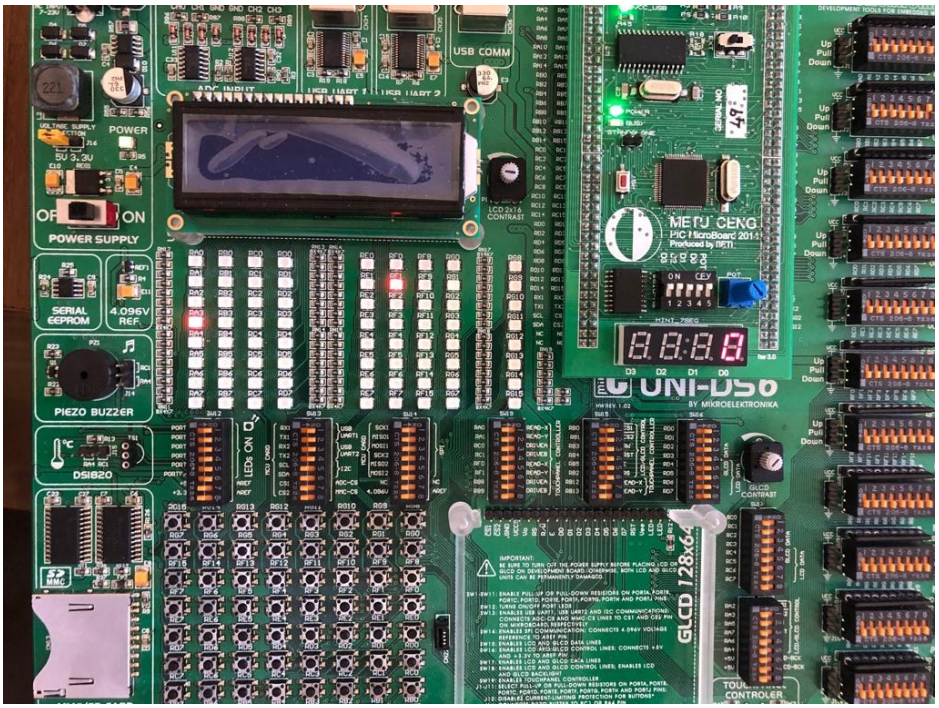

Figure 4: A random asteroid is created on **RF1**.

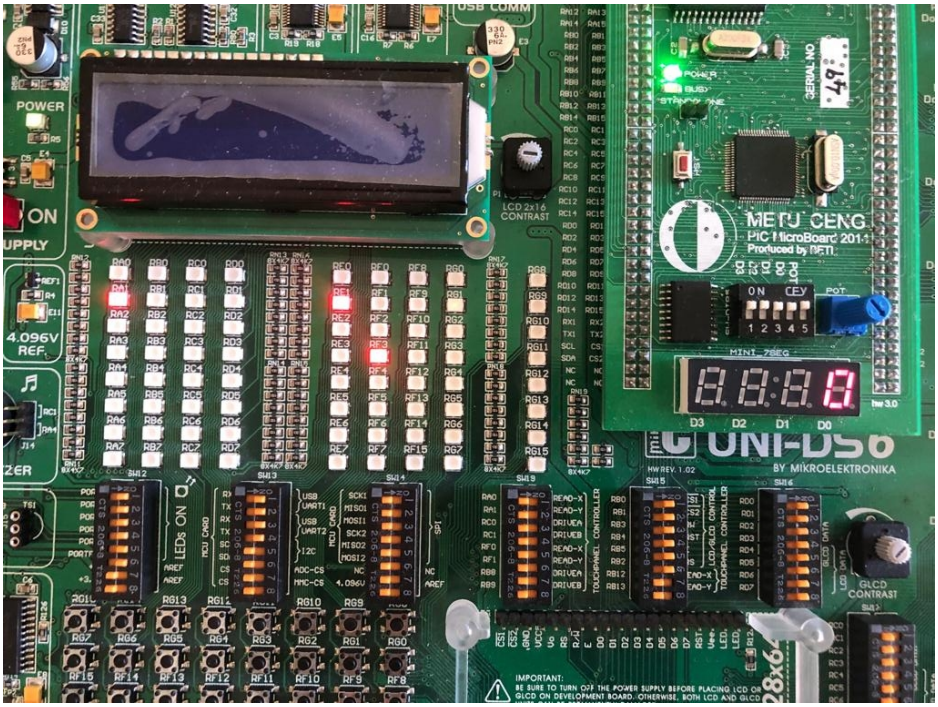

Figure 5: Another random asteroid is created on **RF3**. The first asteroid moved on. The player moved the spaceship up.

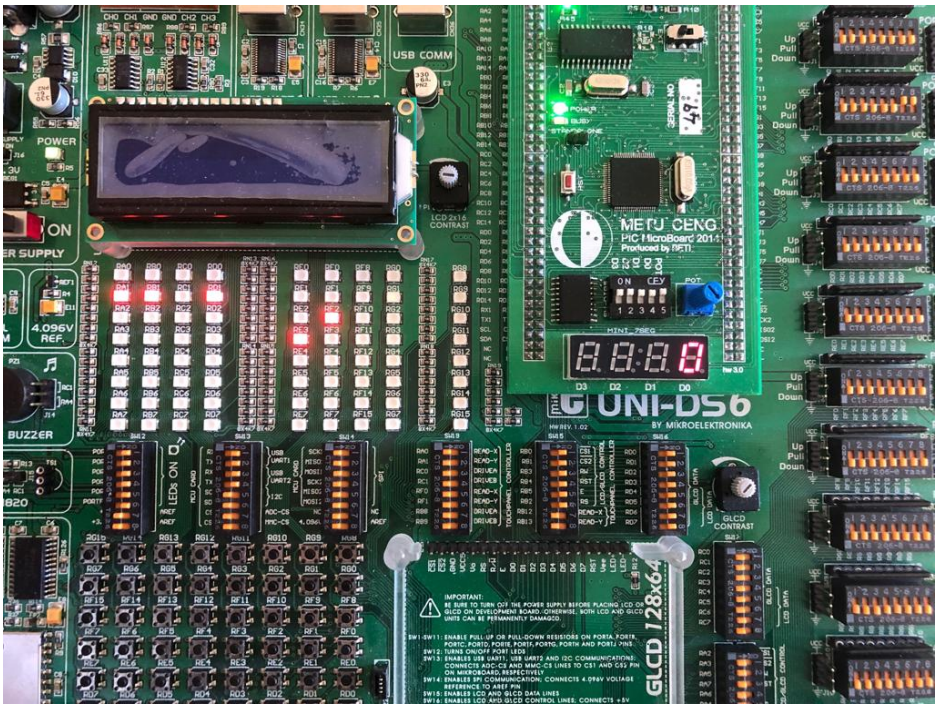

Figure 6: Another random asteroid is created on **RF2**. The other two asteroids moved on. The spaceship fired a laser beam on **RB1**.

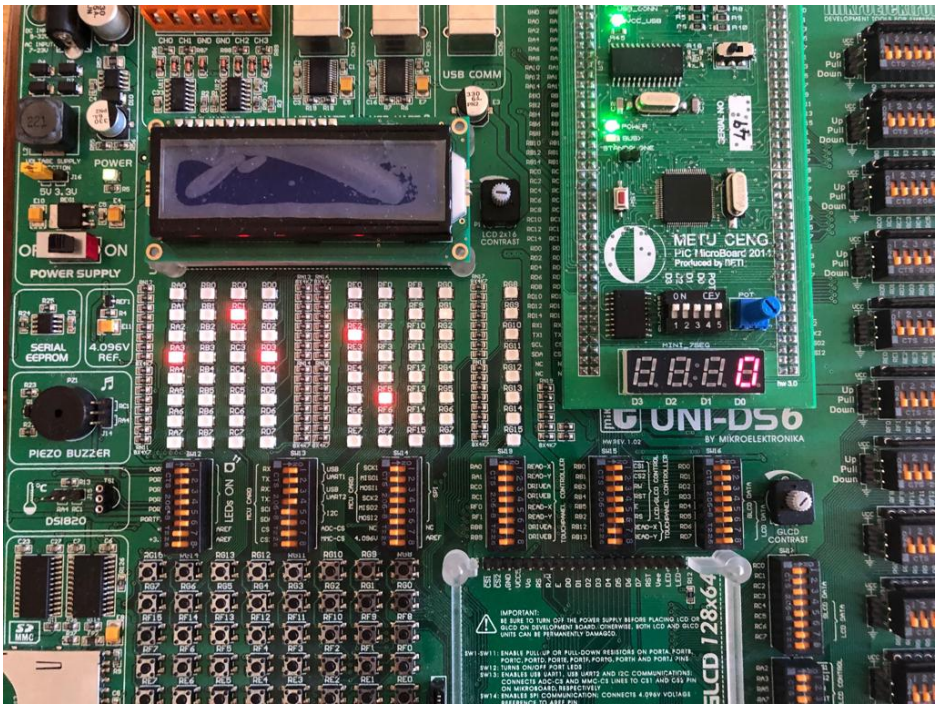

Figure 7: Another random asteroid is created on **RF5**. The laser beam hit the asteroid on **RC1**. The other two asteroids moved on.

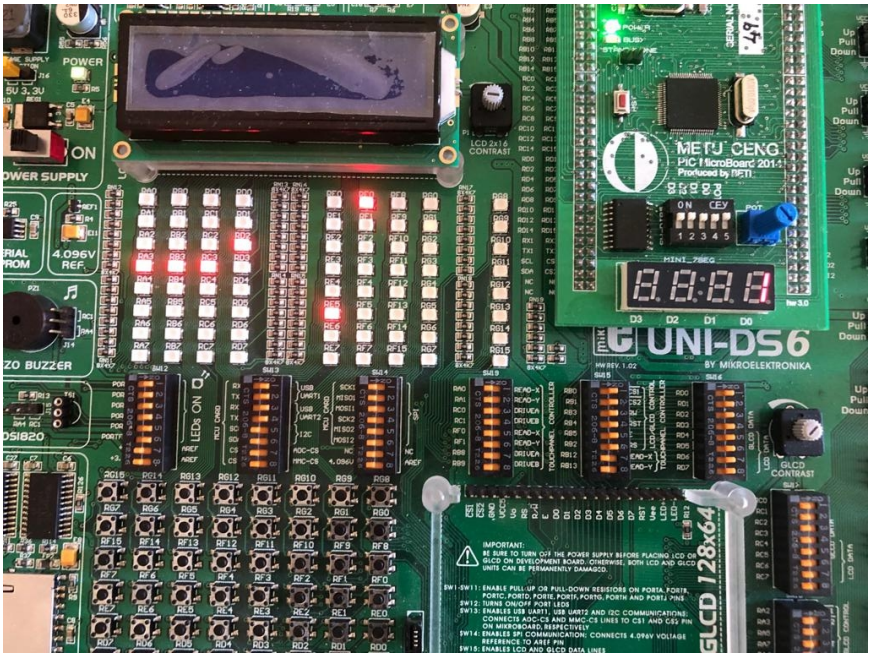

Figure 8: Another random asteroid is created on **RF0**. The player moved the spaceship down and fired a laser beam on **RB3**. The other three asteroids moved on. Note that after the collision on **RC1** from previous step, both laser beam and asteroid are disappeared, and the player got a score point.

3.7 Switch Configuration

You should use the following switch configuration in Figure 9.

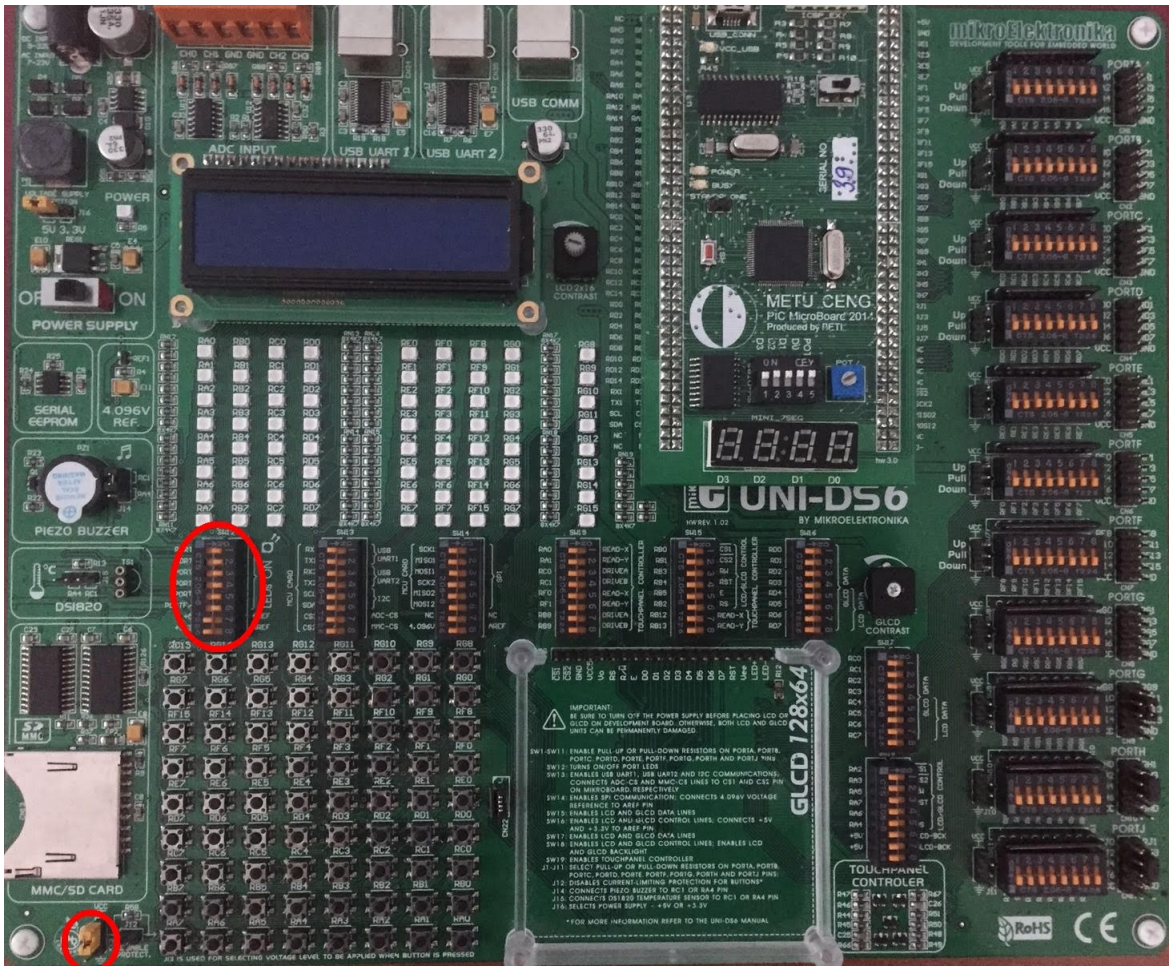

Figure 9: Switch Configuration

4 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 will be used to show spaceship.
- PORTA, PORTB, PORTC, PORTD, PORTE, PORTF will be used to show the asteroids.
- 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 spaceship.
- **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.

5 Hints

5.1 7-Segment Displays

There are four 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 below. Notice that PORTH pins are connected to all displays.

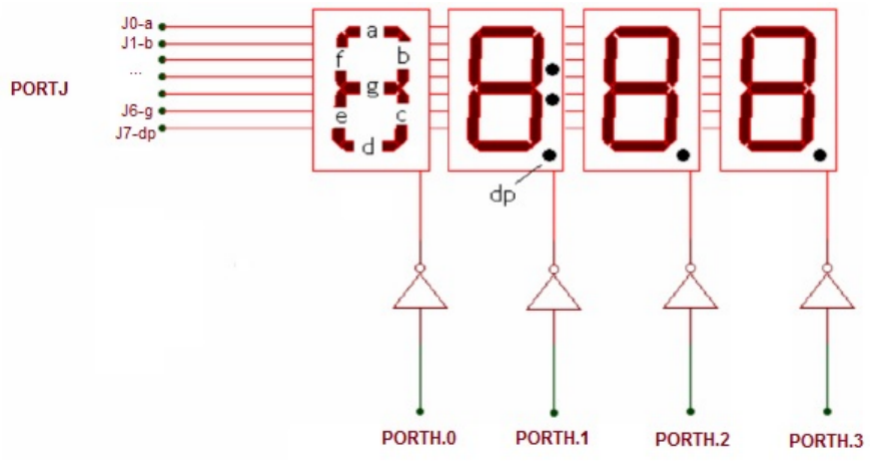

Figure 10: Connections for the 7-Segment displays on the development board.

As an example, if you want to show number "4" on leftmost display (DISP0), 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 DISP0 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 DISP0 (leftmost 7-segment display) and there are 3 dp on DISP1 which run simultaneously.

If you want to show, for instance some value(1234) on the displays, you should select only DISP0 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 DISP1, DISP2 and DISP3. 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.

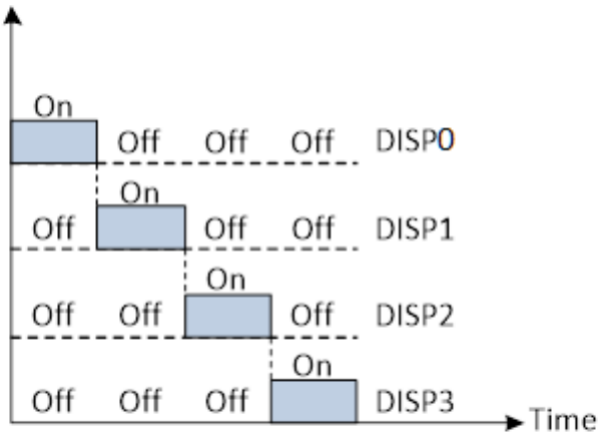

Figure 11: Graphical illustration to show characters on each 7-Segment displays simultaneously.

6 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]