Middle East Technical University **CENG 336**
Computer Engineering Department **Spring 2016**

Take Home Exam III
Revision 1.1
Deadline: 08.05.2016 Friday, 23:55

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

1. Introduction

The purpose of this assignment is to familiarize you with the **ADC module** of the PIC, as well as using the **LCD module** together with **TIMERs** and **INTERRUPTs**. You will implement a game called "Find the Pair". The goal is to find the identical pairs among 10 characters (5 pairs of characters). The characters should be shown to the user initially, and should be hidden later. Then the player chooses a pair of characters, one after another. If the chosen characters are the same, that pair is kept opened and should not be hidden anymore. The game will last in 99 seconds. The user is expected to find all pairs in the given time.

2. Specifications

- **40 MHz** frequency and **10-bit** adjusted ADC will be used. The ADC value will be used to move on the character matrix.

- The ADC value should be 0 when you turn the ADC potentiometer clock-wise to its leftmost position, and it should be 1023 when you turn the ADC potentiometer clock-wise to its rightmost position.

- In this homework, you will use **TIMER0** interrupt with a period of **100ms**. The **Timer0 ISR** should start ADC conversion. You should also use an **ADC ISR** to detect the end of conversion and read the converted value. Consequently, you should be sampling the potentiometer value with a frequency of **10Hz**. Each ADC read should update the active cell (is explained below). **Timer0** should be configured as **8 bit**.

- **When the PIC is powered up**, the first screen should be like in **Figure 1**. The initial screen will be shown for **exactly 3 seconds**. You should use **TIMER1** to produce three second time delay. (You will use **TIMER1 interrupt** with a period of **3000 ms** and the **Timer1 ISR** should control the time.)
-

	F	**i**	**n**	**d**		**t**	**h**	**e**		**P**	**a**	**i**	**r**	**!**	
	*	*	*	*	*	*	*	*	*	*	*	*	*	*	

Figure 1. The first screen when PIC is powered up.

- Then the game will start by showing the character map on LCD module. The characters will be shown to the user for **3 seconds** again, then the characters are hidden. Again, you should use **TIMER1** to produce three second time delay. (You will use **TIMER1 interrupt** with a period of **3000 ms** and the **Timer1 ISR** should control the time.)
- In **Figure 2**, the character map is given. You will use the same character map at each run (also during THE-3).

	!	**?**	**!**	**#**	**=**			**A**	**x**	**i**	**s**	**:**	**X**		
	#	**=**	**;**	**?**	**;**			**R**	**:**	**0**		**C**	**:**	**0**	

Figure 2. Initially, characters are shown to the user.

- After the second 3 s period (6 s later from the initialization), you must hide characters by showing the character '█' (0xFF) instead of actual characters (**Figure 3**). We will use an invisible (so-called) cursor on the character map to indicate current position. You will use ADC module to change that position. *Axis* and *R*/*C* values on the LCD module are used for that purpose.

	█	**█**	**█**	**█**	**█**			**A**	**x**	**i**	**s**	**:**	**X**		
	█	**█**	**█**	**█**	**█**			**R**	**:**	**0**		**C**	**:**	**0**	

Figure 3. LCD screen when characters are hidden.

- When the LCD turns from Figure 2 to Figure 3, at the same time, you should show the time in 7-segment displays D3 and D2 (**Figure 4**). The player is expected to find all pairs in **99 seconds**. The timer starts with the value 99 and decreases this value **once in every second**. You should use **TIMER1** to produce one second time delay. (You will use **TIMER1 interrupt** with a period of **1000 ms** and the **Timer1 ISR** should reduce the time once every second.)
- D1 and D2 show the number of pair that the user found so far. We have totally 5 pairs and the possible values of D1 and D0 are "00", "01", "02", "03", "04", and "05". Initially, D1 and D0 should be "00" as shown in Figure 4.

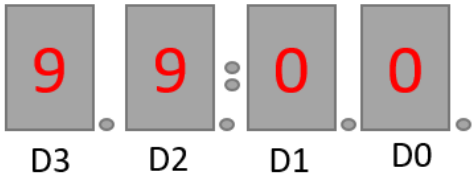

Figure 4. 7-segment Displays when the game is started

- **"Axis: _"** shows the "active axis" (X or Y) in which the player can move the cursor by changing the value of the potentiometer. The **RB6** push button will be used to change the active axis. The default axis should be X (the horizontal axis). When **X** is active, the player can change its position **on the same row** (*R* value remains the same and *C* value changes). If it shows **Y** (the vertical axis), the player can change its position **on the same column** (*C* value remains the same and *R* value changes).

- **"R: _ C: _"** current position of the cursor. It shows us the index values of the matrix that is shown on the LCD module (R for "row" and C for "column"). R value can be "0" or "1", and C value can be a number in 0-4 interval. For example, if player wants to choose the cell in the second row and the third column, R should be 1 and C should be 2.

- As an example, after you read the initial position of the ADC potentiometer, let say the cursor is on the coordinates (0, 0), the top leftmost cell. For this case, you can see the corresponding Axis, and R/C values in Figure 3.

- You must blink a hidden cell when the cursor is on it. Blinking should be in a **250 ms time interval**. In other words, in the first 250 ms, you should show the '▉' character and in the next 250 ms the space character (" ") should be shown. You should use **TIMER1/TIMER0** to produce 250 ms time delay. (You will use **TIMER1/ interrupt** with a period of **250 ms** and the **Timer1/Timer0 ISR** should change the time.)
-

- In **Figure 5**, the player chooses the character in (0, 0) by pressing the **RB7** and the hidden character appears.

	!	▉	▉	▉	▉			**A**	**x**	**i**	**s**	**:**	**X**		
	▉	▉	▉	▉	▉			**R**	**:**	**0**		**C**	**:**	**0**	

Figure 5

- In **Figure 6**, player selects the second character. When player opens two characters, the program should show them for **2 seconds** (when evaluating your programs, we will not try to change the cursor's position during 2 s, or press RB6 and RB7). You should use **TIMER1/TIMER0** to produce two seconds ms time delay. (You will use **TIMER1/ interrupt** with a period of **2000 seconds** and the **Timer1/Timer0 ISR** should change the time.)

	!	▉	▉	▉	▉			**A**	**x**	**i**	**s**	**:**	**X**		
	▉	▉	**;**	▉	▉			**R**	**:**	**1**		**C**	**:**	**2**	

Figure 6

- If the chosen two characters are not the same, they will be hidden again using the character '▉' after two seconds. **Figure 7** shows this case.

	▉	▉	▉	▉	▉			**A**	**x**	**i**	**s**	**:**	**X**		
	▉	▉	▉	▉	▉			**R**	**:**	**1**		**C**	**:**	**2**	

Figure 7

❖ Then, the player opens the character in the cell (1, 1), in **Figure 8**. The player selects the second character in Figure 9. Now, there is one "**found pair**". Found pairs should not be hidden again until the end of the game. Also, assume the player finds this pair when the countdown is "75" seconds. Because the user finds his/her first pair, displays D1 and D0 should be updated and show '01' as depicted in **Figure 10**.

	▮	▮	▮	▮	▮			**A**	**x**	**i**	**s**	**:**	**X**		
	▮	=	▮	▮	▮			**R**	**:**	**1**		**C**	**:**	**1**	

Figure 8

	▮	▮	▮	▮	=			**A**	**x**	**i**	**s**	**:**	**X**		
	▮	=	▮	▮	▮			**R**	**:**	**1**		**C**	**:**	**2**	

Figure 9

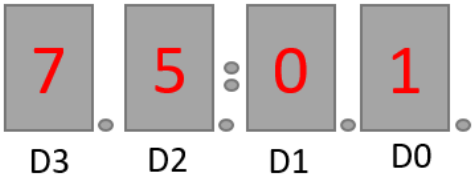

Figure 10

❖ Then the player goes on, and selects another character in Figure 11.

	▮	**?**	▮	▮	**=**			**A**	**x**	**i**	**s**	**:**	**Y**		
	▮	**=**	▮	▮	▮			**R**	**:**	**0**		**C**	**:**	**1**	

Figure 11

❖ If the player can find all pairs before the time expires, the program should show the final LCD screen as shown in **Figure 12**. Remaining time should be shown on displays D3 and D2 (let's say 11 seconds for this case). Moreover, D1 and D0 should show the total number of "found pairs". Final values of 7-segment displays is shown in **Figure 13**.

	!	**?**	**!**	**#**	**=**			**A**	**x**	**i**	**s**	**:**	**X**		
	#	**=**	**;**	**?**	**;**			**R**	**:**	**0**		**C**	**:**	**0**	

Figure 12

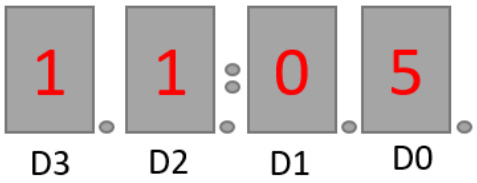

Figure 13

❖ If the player cannot find all pairs before time expires, a possible LCD screen can be like **Figure 14**. In this case, the player finds and three pairs and time expires. Corresponding displays are given in **Figure 15**.

	?		**#**	**=**			**A**	**x**	**i**	**s**	**:**	**X**		
	#	**=**		**?**				**R**	**:**	**0**		**C**	**:**	**4**	

Figure 14

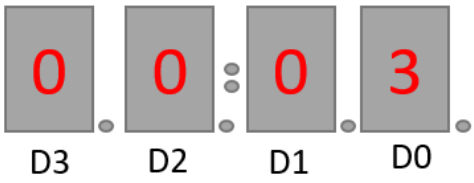

Figure 15

❖ ADC potentiometer values will be mapped to character map rows and columns, according to the table below:

Row/Column	ADC range	Index on the character map
Row	$0 \leq X < 512$	0
Row	$512 \leq X < 1024$	1
Column	$0 \leq X < 205$	0
Column	$205 \leq X < 410$	1
Column	$410 \leq X < 615$	2
Column	$615 \leq X < 820$	3
Column	$820 \leq X < 1024$	4

Table 1: ADC Value to Row/Column Mapping Table

❖ When the cursor is on a cell which is hidden with the '▉' character, then it should blink. However, when the cursor is on an already opened cell (that cell and its pair have been found before), you don't blink that character. This time user keeps track of curser's position with the help of *R*/*C* coordinates.

❖ **RB7** has no effect on an "already opened cell". For example, if the cursor is on '!' and you press RB7, nothing changes. RB7 opens the hidden character (e.g. turns '▉' into '!').

Coding Rules

- You will code your program using PIC C language. You should use **XC8 C compiler** for MPLAB X. You can find the related documents on the recitation 7 and COW.
- Your program should be written for PIC18F8722 working at 40 MHz
- **When the system is started, the LCD module must be all clear, with the configuration that the cursor and blink are off.**
- **You should control RB6 and RB7 buttons with PORTB change interrupt. The pull-up enabling or disabling (using INTCON2bits.NOT_RBPU) cases are up to your design.**
- **You should obey the specifications above about TIMER0 and TIMER1 interrupt implementations.**
- The corresponded switch configuration is also included in THE3 files on COW.
- You can modify the LCD module codes if you need.

Resources

→PIC 18F8722 Datasheet
→The LCD Module Datasheet
→ PIC Development Tool Programming & User Manual
→ Course web page and newsgroup
→ Each related recitation demo codes and notes can be used.
→ LCD example and a template code in Recitation 07 documents on COW.

HOW TO ADD XC8 COMPILER TO MPLAB X IDE ON INEK MACHINES

1. Type "mplab_ide" to open the Mplab X IDE on terminal.

2. `Look at Tool > Options > Embedded` menu. The xc8 compiler is already installed.

3. Select `File -> New -> Standalone Project`. Create new project with the following configurations.
   - Select device -> PIC18F8722
   - Select tool -> PICKIT2:SN BETI
   - Select Compiler -> XC8

Hand In Instructions

→ You should submit your code as a single zip file named as the3_##.zip through COW (## represents your group number). This file will include the3.c, lcd.c, Includes.h and lcd.h. At the top of the3.c, you should write your ID, name and surname as commented out. Do not forget replace ## with your group number.
→ **You should add comments on specific codes or code blocks.**

Grading

Total of the take home exam worth 100 points. You should include brief but descriptive comments in your code, including a larger comment in the beginning of your _le describing the structure of your program with subroutine descriptions (For example the place, function name of your Timer0 ISR, random number generator function etc.) and their relations to one another. Your grade will also depends on the quality of your design, not just its functional correctness. If there is an unclear part in your code, we may ask any of the group member to describe that code segment. Also, group members may get di_erent grades. We reserve the right to evaluate some or all of the groups to determine the contribution of each group member to the assignment.

→ A correct implementation of the timer ISRs and proper timing… Related code and the resulting values will be checked,
→ Proper ADC operation, ADC interrupt implementation and related code will be checked,
→ Correct and problem-free use of LCD functionality,
→ Proper button use and correct implementation of PORTB change interrupt,
→ Proper usage of the 7 Segment Displays,
→ Proper operation of your program.

will be considered while grading.

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]