Middle East Technical University **CENG 336**
Department of Computer Engineering **Spring 2019**

Take Home Exam III

Revision 1.1
Deadline: 30.04.2019, 23:55
Late Submission is not allowed.
Any clarifications and revisions to the assignment will be posted to the ODTUCLASS discussion forum.

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 an electronic voting system, *Electro Vote*, which voters can see candidates on the LCD module and vote a candidate by pushing a button. To scroll up and down candidates on the list, voters use the potentiometer together with ADC module. The most voted candidate is shown on 7-segment displays dynamically; they are updated during the voting period.

When the PIC is powered up, the initial screen on the LCD module should be as shown in **Figure 1**. If the user **pushes and releases the RE1 push button,** then the LCD module should show the screen like in **Figure 1** for **approximately 3 seconds (+/- 5 ms can be acceptable as error range)**. For only this particular 3 second wait, **you can use busy-wait "loops" instead of a timer (you are, however, not allowed to use built-in functions for this task)**. During this time interval, 7-segment displays should remain unchanged.

	#	**E**	**l**	**e**	**c**	**t**	**r**	**o**		**V**	**o**	**t**	**e**	**#**	
	#	**#**	**#**	**#**	**#**	**#**	**#**	**#**	**#**	**#**	**#**	**#**	**#**	**#**	

Figure 1. The first screen when PIC is powered up

After 3-second wait of the initial screen, the authorities enter the total number of potential voters to the system (for a maximum of 99). **Figure 2** shows the LCD module for this case. This number is entered to the 2nd and 3rd cells of the second row, corresponding to tens and units digits, respectively. Initially, there is '0' in these cells. You will use RB6 push button to increment the tens digit first (the 2nd cell). Each press of RB6 increments the digit by one. If the cell contains 9 and the user pushes RB6 again, the digits become 0 again. To enter the number for the tens digit RB7 push button is used. After using RB7, the 3rd cell of the LCD becomes active. This time, each press of the RB6 button increases the units digit. **You should control RB6 and RB7 buttons with PORTB change interrupt.**

	#	**E**	**n**	**t**	**e**	**r**		**V**	**o**	**t**	**e**	**r**	**s**	**#**	
	0	**0**													

Figure 2. LCD module before entering the number of voters.

When the total number of voters is ready, the user presses RB7 again. In **Figure 3** the total number of voters is provided as 28. This second press of RB7 initiates the voting period.

	#	**E**	**n**	**t**	**e**	**r**		**V**	**o**	**t**	**e**	**r**	**s**	**#**	
	2	**8**													

Figure 3. LCD module when number of voters are provided as 28.

When the voting begins, the first row of the LCD module shows the remaining time for voting. Candidates appear at the second line, one at a time, based on the ADC value. Each candidate has an ID and a name.

Figure 4 shows the LCD module when voting begins. The first row contains the "Time left :" description and the remaining time. The voting lasts 90 seconds. The counter for the remaining time should be decremented **exactly** once every second (use **TIMER1** and interrupts to produce this one second time delay). You should use the same positions for the remaining time. Put a "0" to the tens digit when the counter is less than 10 (09, 08, and so on).

		T	**i**	**m**	**e**		**l**	**e**	**f**	**t**		**:**	**9**	**0**	
	>	**0**	**2**		**Z**	**w**	**e**	**i**	**g**			**:**	**0**	**0**	

Figure 4. LCD module when the voting begins.

The second row of the LCD module contains a '>' character. This character should blink during the voting period. The blinking should be done with a **250 ms time interval**. In other words, in the first 250 ms, the '>' character should be shown and in the next 250 ms, the space character (' ') should be shown. You should use **TIMER0** and the associated interrupt to produce the required exact 250 ms time delay.

After the '>' character, the candidate's ID and the name are shown at the second row. The candidates are shown in the LCD module based on the ADC value. **Table 1** shows the IDs, names, and corresponding ADC values for candidates. For **Figure 4** it is assumed that the initial value of the ADC is 700. Therefore, you should obtain the ADC value at the beginning of the voting. ADC potentiometer is used to change the candidate on the LCD module.

After the name of the candidate, current votes of her/him is shown (which is initially zero). The votes are also shown with two digits. The maximum number of votes can be 99 for a candidate.

For the first row of the LCD module, only the counter on the 14th and 15th cells are updated during the voting period. The positions of the data at the second row should be as follows:

- The 1st cell should be empty.
- The '>' character is placed to the 2nd cell. Remember that it blinks during the 90-second voting period and indicates the current candidate to be voted.
- The ID of the current candidate is shown in the 3rd and 4th cells.
- The 5th cell should be empty.
- Candidate's name is placed between the 6th and 11th cells, left justified.
- The 12th cell should be empty.
- The 13th cell contains the ':' character.
- The vote counter is placed to 14th and 15th cells
- The 16th cell should be empty.

Table 1. Mapping of ADC Values to candidates.

ID	Candidate	ADC Value
18	Poe	0-127
64	Galib	128-255
34	Selimi	256-383
23	Nesimi	384-511
33	Hatayi	512-639
67	Zweig	640-767
63	Nabi	768-895
99	Austen	896-1023

The '>' character indicates the candidate to be voted. To vote for the current candidate, **RB7 button** is used. When RB7 is pressed and released, the counter of the current candidate is incremented by one.

7-segment displays show the most voted candidate after the first vote. Until the first vote is given, the states of the 7-segment should be as shown in **Figure 5**. In case of a tie, do not update the 7-segment. For example, at the beginning of the voting (when all vote counters are zero), a voter vote for Poe. The second row of the LCD becomes "18 Poe :01". Because Poe is the current leader, 7-segment displays are updated as shown in **Figure 6**. Then, another voters votes for Hatayi and the counter for him becomes "01". At this moment, there are two candidates with one vote. Our election rules ignore the tie case: to become the leader you must beat the current leader. So, 7-segment displays remain the same.

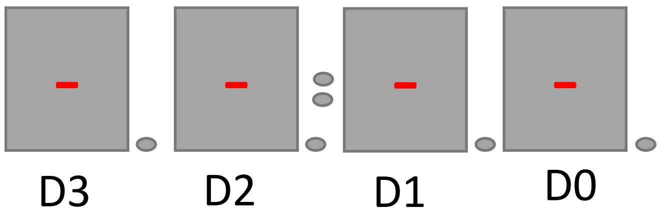

Figure 5. 7-Segment displays when there is no vote.

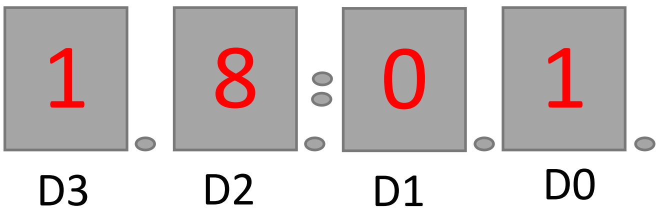

Figure 6. 7-Segment displays showing the leader after voting begins.

When the 90-second voting period ends:

- Time left should be "00".
- ADC potentiometer should be still active to allow scroll up and down.
- The '>' character at the second row stops blinking.
- 7-segment displays should keep showing the leader.
- The user resets the pic for new voting.

Figure 7 shows the state of the LCD module and 7-segment displays after the 90-second voting period ends (current ADC value is 500).

		T	**i**	**m**	**e**		**l**	**e**	**f**	**t**		**:**	**0**	**0**	
	>	**2**	**3**		**N**	**e**	**s**	**i**	**m**	**i**		**:**	**1**	**5**	

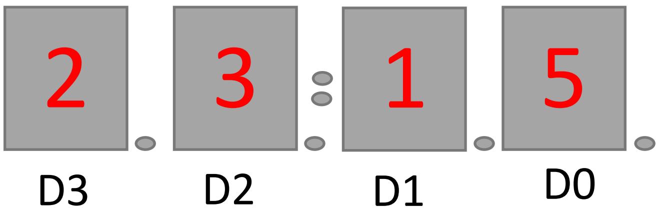

Figure 7. LCD module and 7-Segment displays after the deadline.

The voting can end before the 90-second period ends if the total number of votes is equal to the number of potential voters (provided by authorities). The only difference between this case and the above case is that time left should show the remaining time. **Figure 8** shows the state of the LCD module and 7-segment displays after the voting is ended when all voters are voted before the 90-second period ends (current ADC value is 100, the leader is 23).

		T	**i**	**m**	**e**		**l**	**e**	**f**	**t**		**:**	**2**	**5**	
	>	**1**	**8**		**P**	**o**	**e**					**:**	**1**	**2**	

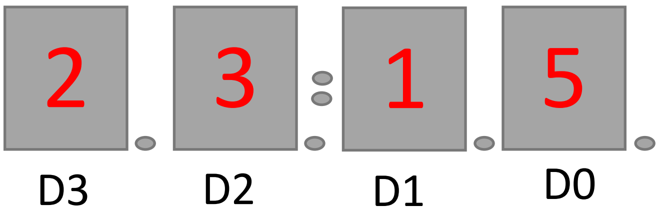

Figure 8. LCD module and 7-Segment displays when all of the voters vote before the deadline.

2. Specifications

- ❖ **40 MHz** frequency and **10-bit** adjusted ADC will be used. The ADC value will be used to cycle through product categories and lists.

- ❖ 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 exam, you will use **TIMER0** interrupt with a period of **100 ms**. 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 **10 Hz**. Each ADC read should update the active LCD module line (explained below). **Timer0** should be configured to be **8-bit**.

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 Recitation04 and ODTUCLASS.
- ▪ 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 rules corresponded with the push button connected to RE1 pin.**
- ▪ **You should obey the specifications above about TIMER0 and TIMER1 interrupt implementations, and also the busy wait loop specification.**
- ▪ **The corresponded switch configuration is also included in THE3 files on ODTUCLASS. If you have a different configuration, it is possible, upload also this configuration figure with showing the differences.**
- ▪ You can modify the LCD module codes if you need for latency or etc...
- ▪ **Usage of any push button or the potentiometer should not cause flicker/loss on the LCD module and 7-segment displays.**

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.
→For 7-segment displays usage you can look at the Section 5.1 of THE2.
→ LCD example and a template code in Recitation5 documents on ODTUCLASS.

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 / PICKIT3
   - Select Compiler -> XC8

Hand in Instructions

→ You should submit your code as a single zip file named as the3_##.zip through ODTUCLASS (## 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 in “the3.c” file. Do not forget replace ## with your group number. **Only one group member must submit the the3_##.zip by considering the naming issue as before, otherwise you will lose partial points.**
→ **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 depend 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 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.

→ 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]