Middle East Technical University Department of Computer Engineering

CENG 336

Introduction to Embedded Systems Development
Spring 2020
THE2

Due date: 10.05.2021, Monday, 23:55

1 Objectives

This assignment covers basic input/output operations together with programming with interrupts and timers. This will be done in the context of implementing an old school mobile phone (see Figure 1) keypad for text input.

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: Old phones with keypads used this text input scheme.

2 Specifications

2.1 Scenario

The primary characteristic of old school mobile phones were how they encoded 26 letters with 9 inputs of the keypad. The resulting system for this assignment should allow character input using the 4x3 Keypad located on the PICGenios board for PICSimLab (see Figure 2). We will use the ASCII Turkish character set; A to Z without Q, W, X or Ç, Ğ, İ, Ş, Ü, Ö. This gives us 24 characters to work with plus the space character.

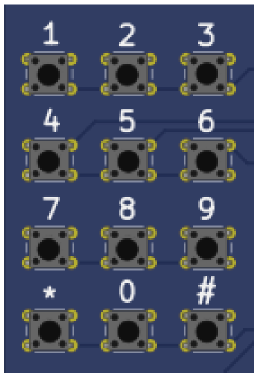


Figure 2: Keypad located on the board used for text input.

The individual letters are input using multiple key presses, for instance ’A’ is input using only one key press on button ’2’ but ’F’ requires 3 key presses in quick succession on button ’3’. The input letters will be shown in the 7-segment display using an appropriate 7-segment font. A real 7-Segment font is very constrained and somewhat unreadable, so the “font” will be shared with you in the *Encoding* section.

The system will be in one of the three states at any given time; *Message Write*, *Message Review* or *Message Read*.

Using the **RB4 button interrupt** will immediately switch the system between Message Write State and Message Review State.

After all 6 characters are entered in Message Write State or when countdown reaches to 0 at Message Write State or Message Review State, the system will enter the Message Read State.

2.2 Initial Configuration

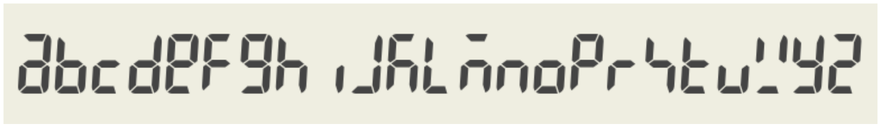
Initially, the system should be waiting for the **RB3** button action. At this stage, 7 segment display screen should be blank. After pressing & releasing the **RB3** button, the system should be brought to *Message Write* state and the countdown should start.

2.3 Encoding

With the provided 7-Segment font, your alphabet will look like the following.

1	2 ABC	3 DEF
4 GHI	5 JKL	6 MNO
7 PRS	8 TUV	9 YZ␣
*	0	#

Table 1: Input encoding overview



(a) Alphabet used in the THE with the 7-Segment Font

abcdefghijklmnoprstuvyz

(b) Reference letters

Figure 3: How your message will look in the 7-Segment display

2.4 Message Write State

This state accepts inputs through the keypad and displays the countdown, last letter input and current input.

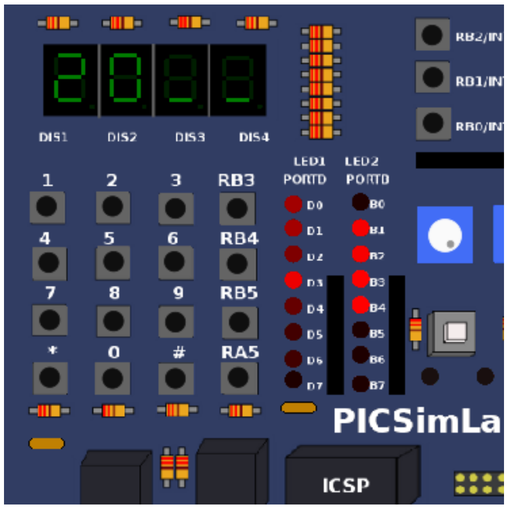


Figure 4: 7-Segment Display after RB3 button action

Going from left to right; leftmost two displays (DIS1 and DIS2, see Hints section) will be used for displaying the countdown. The countdown should start from 20 seconds and should be updated by **TIMER0 Interrupt**. The next 7-segment display will display the last letter input and the rightmost one will display the current input. The key presses should immediately affect the current input.

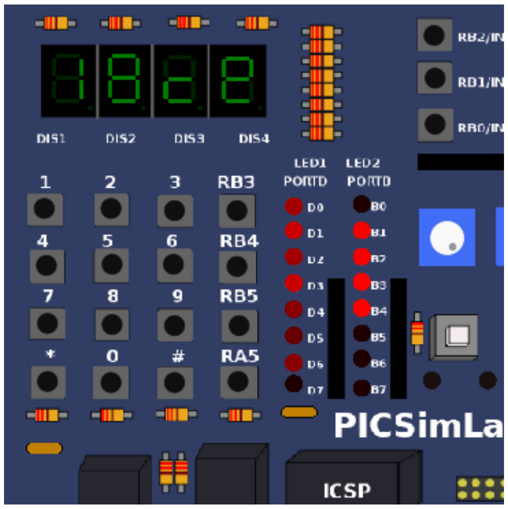


Figure 5: 7-Segment Display after pressing '2' for 3 times and '3' for once on keypad

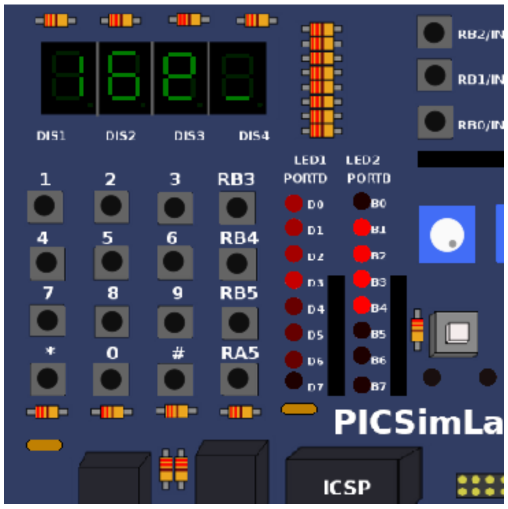


Figure 6: 7-Segment Display after waiting for 1 second to commit the character and switch to the next one during the Message Write State

Table 1 shows which letter is accessible through which button. Every button can input 3 characters, each push and release on the same button cycles through the available characters under that button. 4th press will cycle back to the first letter. See Table 2 for an example.

If the **RB4** button interrupt occurs during Message Write State then the system switches to Message Review State. This state will also immediately

Times	1st	2nd	3rd	4th	5th	...
Character	A	B	C	A	B	...

Table 2: Required behaviour when button 2 is pressed multiple times

switch to Message Read State, if all 6 characters have been input or the global timeout duration is reached.

- Any empty positions are shown as "_". At the start, all of the letters in the message text are shown on the 7 segment display as "_".
- There is a space character which is accessible through the button '9'. Contrary to the empty positions, if a space character is entered, none of the segments on that 7-segment display will be on (see Figure 8).
- Pressing '1', '0', '*' or '#' on the keypad will have no effect at this state.
- The current letter input is saved after 1 second of inactivity (see Figure 6) or when a new letter is input with a different button.
- To input two consecutive letters encoded with the same button, you have to wait for the 1 second timeout duration.
- The **RB4** interrupt *will discard* the current character as well if the 1 second timeout period has not been hit yet. So any incorrect letter input can be avoided with switching to Message Review State by pressing and releasing RB4 button before the 1 second period.

2.5 Message Review State

In this state, you can review the message by going back and forth using over the letters with "*" and "#" buttons on the keypad.

Initially, the first 4 letters will be displayed on the 7-segment display screen. "*" and "#" buttons will be used to scroll left and right respectively. At each button press and release, the text on the display will scroll left or right by **one letter only**.

If an **RB4** button interrupt occurs during this state, the system switches to Message Write State.

This state will immediately switch to Message Read State, if the global timeout duration is reached.

- The other keypad buttons have no effect at this state.
- If all 6 letters have not been entered yet, the blank positions will be shown with "_" on the 7-Segment display (see Figure 8).
- If you have reached the end of the message, "#" button will have no effect and accordingly, if you have reached the start of the message the "*" button will have no effect on the 7-segment displays.

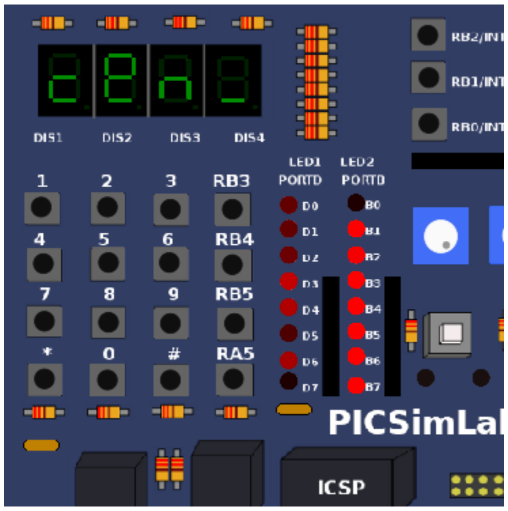


Figure 7: 7-Segment Display in Message Review State after inputting 3 letters

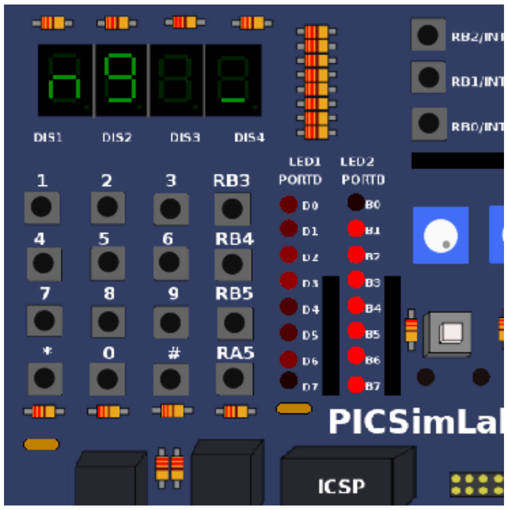


Figure 8: 7-Segment Display in Message Review State after pressing '#' twice

2.6 Message Read State

In this state, the final message text will start scrolling horizontally on the 7-segment display. The system should be unresponsive to any button action. It will start by displaying the first 4 letters. Every half a second, it will introduce the next letter by scrolling left. Once it reaches the 6th letter, it will start scrolling right. This behaviour continues until powering off the PICGenios board.

3 Regulations and Submission

- You will code your program using PIC assembly language.

- Your program should be written for PIC18F4620 working at 40 MHz.
- **You MUST properly comment your code, including a descriptive and informative comment at the beginning of your code explaining your design.** %5 of your grade will depend on the quality of your comments.
- You will use 3 ports in this assignment; PORTA, PORTB, PORTD.
- PORTB and PORTD will be used to input letters through multiple keypresses. There is a detailed explanation on how to read input from the keypad on Hints section.
- PORTA and PORTD will be used as output ports to display the resulting message on 7-Segment displays. You can find some useful information on Hints section (4).
- You should use the Timer0 interrupt as stated above and configure the Timer0 to work in 16-bit mode.
- You should use PORTB button interrupts for implementing the RB4 button action. The specified behaviour should occur after releasing the RB4 button.
- It is beneficial to avoid function calls in 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.
- 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. Please consult these resources first before you ask any questions to the assistants or on the forums.
- There is a discussion forum already setup for THE2 for your questions, you can also ask specific questions through mail to either Merve or Yigit.
- The final `.asm` file will be submitted through ODTUClass.

4 Hints

4.1 7-Segment Displays

There are four common cathode 7-segment displays mounted on the PIC-Genios board. PORTD and PORTA are used as data bus and selection pins, respectively, as illustrated in Figure 9. Notice that PORTD pins are connected

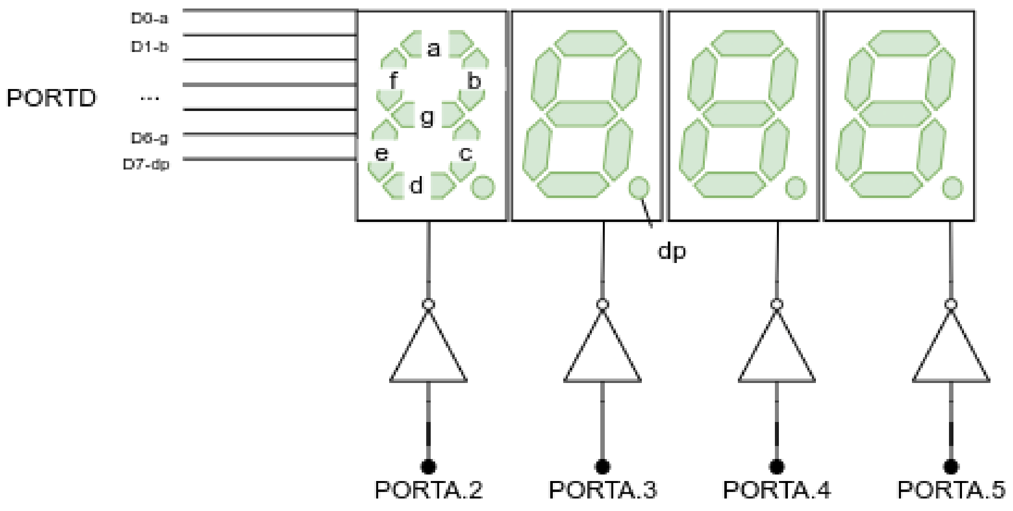


Figure 9: 7-Segment Display Reference

to all displays. As an example, if you want to show number 4 on leftmost display (DIS1), you should first select it by setting RA2 and clearing RA3, RA4 and RA5, then send binary 01100110 to PORTD. Hence, a, d, e segments on DIS1 will be turned off, and b, c, f, g segments will be turned on. Note that RD7 pin is used for the decimal point (dp) of displays.

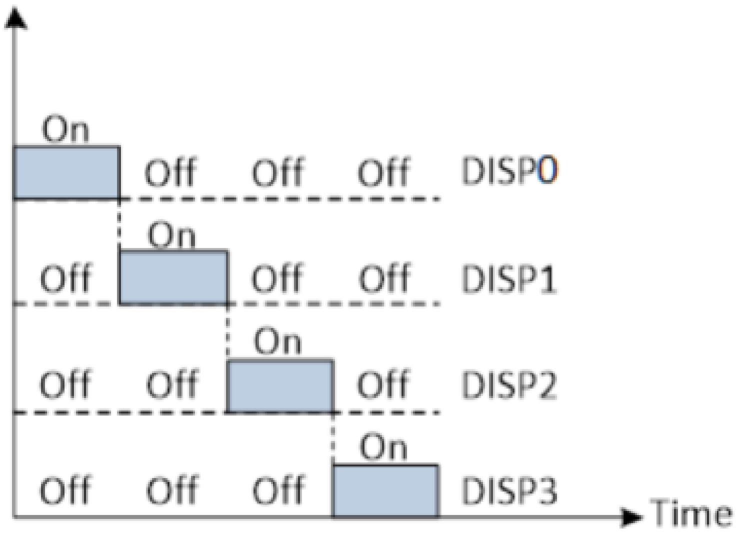


Figure 10: Graphical illustration to show characters 7-Segment displays simultaneously.

If you want to show, for instance some value (1234) on the displays, you should select only DIS1 by using PORTA, write the byte necessary to turn on segments to PORTD, and wait for a while. Then you should do the same for DIS2, DIS3 and DIS4. This is illustrated in Figure 10. 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.

4.2 Keypad

We will use 4x3 Keypad located on the PICGenios board. There are 7 pins connected to the keypad but 12 buttons to interact with. See Figure 11.

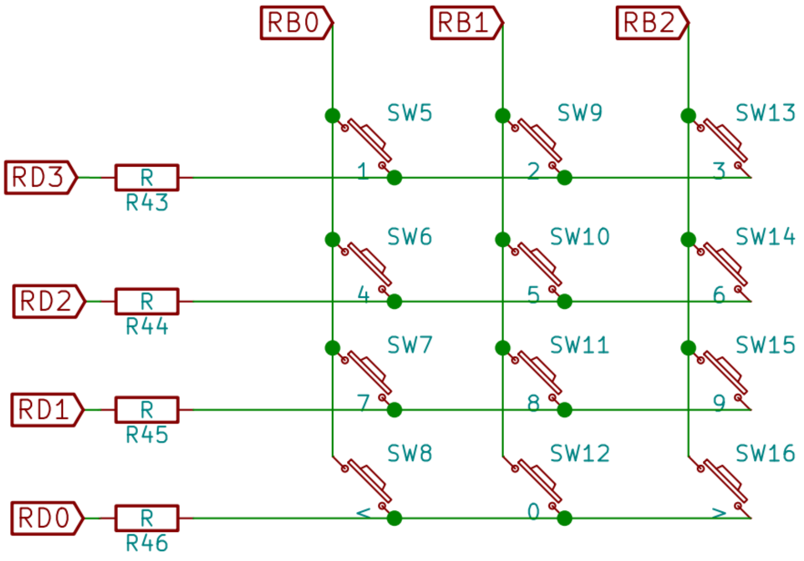


Figure 11: Keypad Reference

In order to use the keypad the pins connected to the columns of the keypad matrix should be set as output and the pins connected to the rows of the keypad matrix should be set as input. When a key press occurs on the keypad, the relevant pin on the row will set LOW if the relevant pin on the column is LOW. So by setting the column pins LOW in turns and checking the input from row pins, we can match the key being pressed.

5 Cheating

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

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 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 or code directly from someone else - whether copying files or typing from someone elses 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 elses 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