Middle East Technical University Department of Computer Engineering

CENG 336

Introduction to Embedded Systems Development
Spring 2022
THE2 – Interrupts and Timers

Due: 12. May 2022 23:55
Submission: **via ODTUClass**

1 Objectives

THE2 aims to familiarize you with the basic input/output operations together with programming with interrupts and timers in C. This will be done in the context of implementing a simple version of **Guitar Hero** (see Figure 1) video game. It is a group assignment. Any clarifications and revisions related to the assignment will be posted on **ODTUClass**. You can ask your questions in the discussion forum dedicated for this assignment in **ODTUClass**.

Figure 1: A screenshot from Guitar Hero 3 video game

2 Scenario

Guitar Hero is a well-known series of music rhythm video games used with a guitar-shaped controller. The main purpose of the game is to match the notes that scroll on-screen to coloured circles by pressing the buttons in the controller when the notes is on the coloured circles. In context of the game scenario we are expecting you to implement a basic version of this game.(We do not have a guitar-shaped controller, but we have some buttons on the development board.)

In the game scenario, notes will be created continuously and will slide from left to right. The player should match the sliding notes when they are in the matching circles by pressing the appropriate button on the development board. The game consists of **three levels** that all of them have a different difficulty level. The player starts with **9 health points**. In every missing or wrong notes the player will lose one health point. You will display the current level and remaining health points on seven segment display screen. When the player has no health point, then the game is over. You will use the **Timer0 Interrupts** to create notes continuously and you will use the on-board buttons to match the notes. Also, as a screen, you will use on-board LEDs connected with ports.

3 Specifications

3.1 Initial Configuration

The game will be played in the area of the on-board LEDs connected with RA0-RA4, RB0-RB4, RC0-RC4, RD0-RD4, RE0-RE4, RF0-RF4. Figure 2 shows the game area where the notes slides from left to right. The notes will move across these LEDs. The game level and remaining health (initially 1 and 9) can be seen in the seven segment display. The first digit of the seven segment display shows the remaining health points and the last digit of the seven segment display shows the current level. The game should start as soon as **RC0** button is pressed.

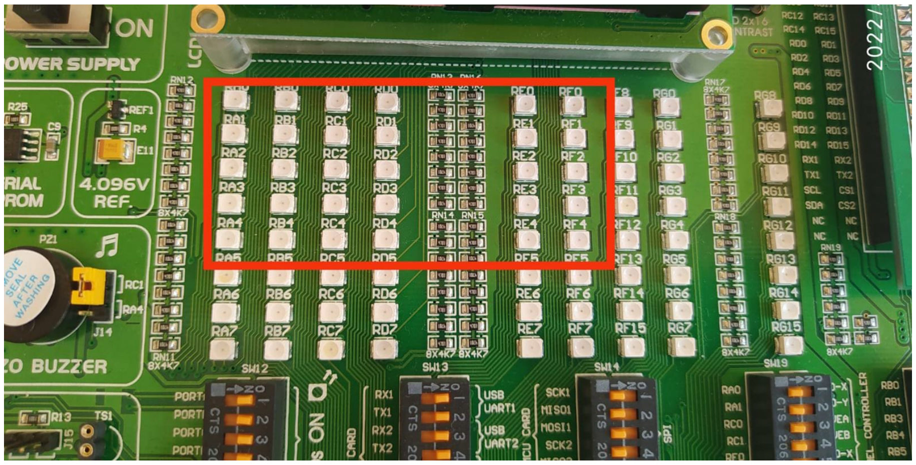

Figure 2: Pre-defined game area.

3.2 Generation and Movement of the Notes

- Each notes should be created randomly on one of the LEDs associated with PORTA. (For random creation, you can find an useful algorithm in Hints section.) In the real Guitar Hero game, multiple notes can be created on multiple locations but in our basic implementation one note per creation is enough.
- The notes should move horizontally. For example, let's say that a note is created in RA0 LED. Then, next location of that note is RB0. The full path of the note is RA0, RB0, RC0, RD0, RE0 and RF0. After RF0, the note should be disappeared.
- The notes should move to their next location with proper timing according to the level.
  - **Level 1**
    - You should create a new note in every 500 ms.
    - You should move the current notes to their next location in every 500 ms.
    - You should create 5 notes at this level.
    - After the creation of 5 notes, the game should pass to the next level.
  - **Level 2**
    - You should create a new note in every 400 ms.
    - You should move the current notes to their next location in every 400 ms.
    - You should create 10 notes at this level.
    - After the creation of 10 notes, the game should pass to the next level.
  - **Level 3**
    - You should create a new note in every 300 ms.
    - You should move the current notes to their next location in every 300 ms.
    - You should create 15 notes at this level.
    - After the creation of 15 notes, the game should end.
- **You should use Timer0 interrupts to create time intervals.**
- You can find some pictures from the game related with different states in Figures 3, 4, 5

3.3 Matching Notes to Circles

- In our implementation, the circles are placed in the LEDs associated with PORTF. That means the player should press the appropriate button when the note is in RFX.
- As input buttons, you will use RG0-RG4. Each button is associated horizontally. For example, when the note is in RF4, the player should press RG4 button to match the note to the circle.
- When the button is pressed on, the note in RFX should disappear.

- If the player misses a note or presses on a wrong button, the player will lose one health point.

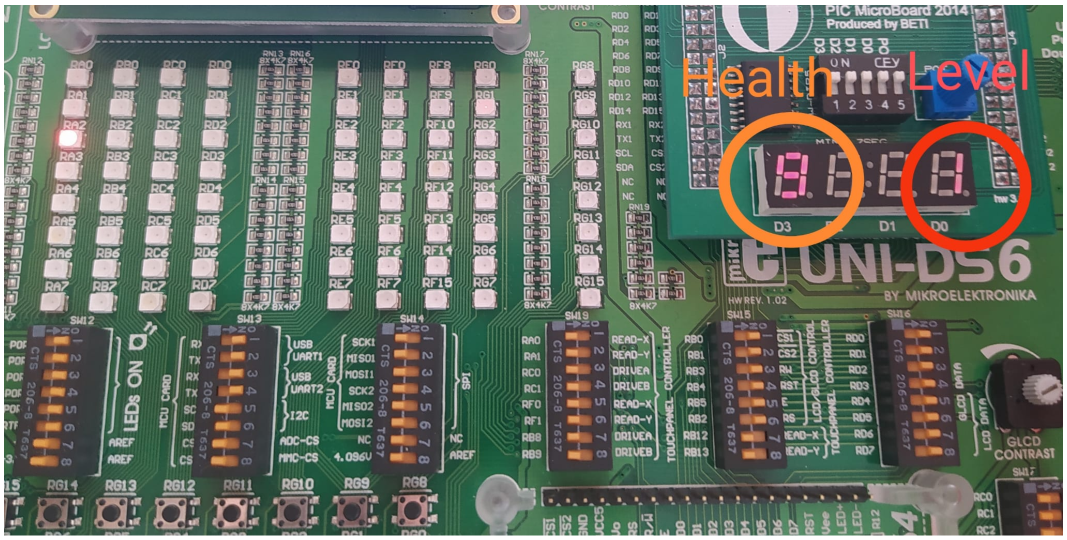

Figure 3: First random note generation.

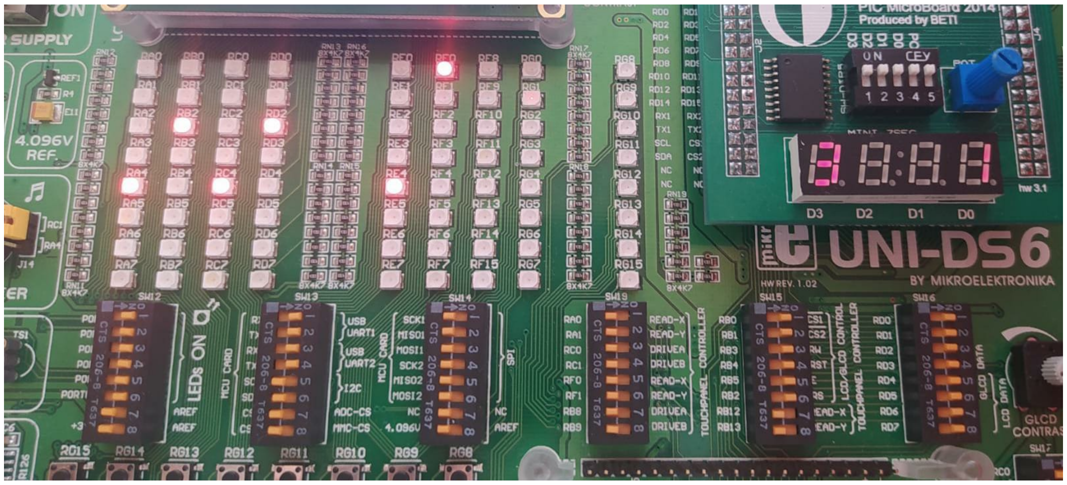

Figure 4: An example picture from Level-1 with 3 remaining health points.

3.4 End of the Game

The game can be over due to two different reasons. The first one is that the health point of the player is zero. When the game is over for this reason, the player ideally should see **LOSE** at the seven segment display. The second one is the player successfully matches all the notes.(30 notes) When the game is over for this reason, the player ideally should see **End** at the seven segment display(Figure 5). The game waits the player to press on RC0 button. When RC0 button is pressed on, the game starts again.

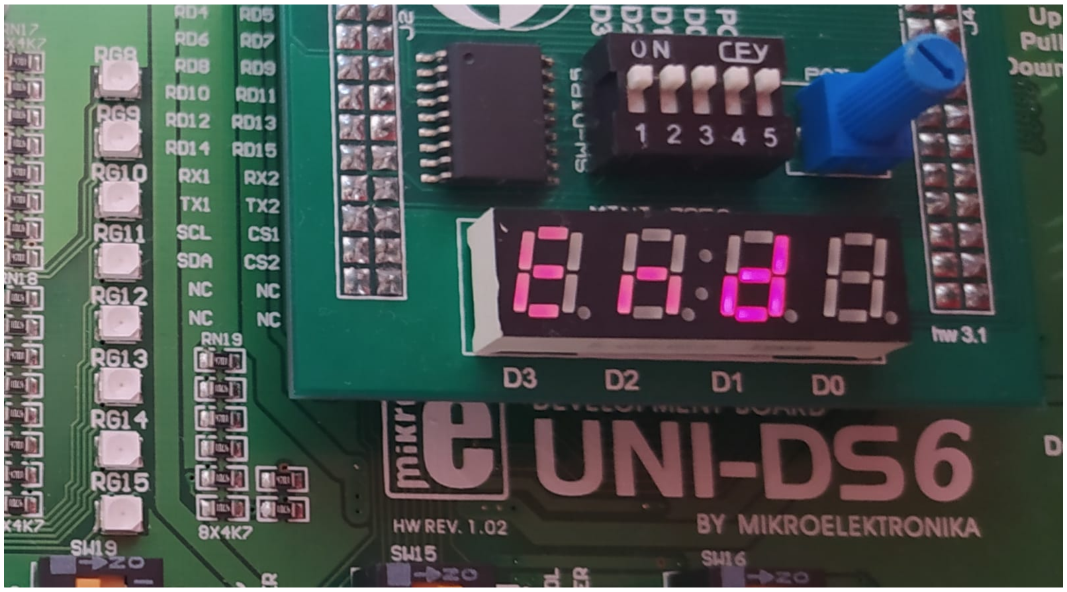

Figure 5: An example game end.

4 Implementation and Regulations

- You will code your game by using C.
- Your program should be written for PIC18F8722 working at **40 MHz**.
- You will work with your groups.
- You will use nine different ports for this assignment: PORTA, PORTB, PORTC, PORTD, PORTE, PORTF, PORTG, PORTH, PORTJ.
- PORTA, PORTB, PORTC, PORTD, PORTE, PORTF will used for the game area.
- PORTG will used for matching the notes.
- PORTH and PORTJ will be used for seven segment display. You can find useful information on Hints section.
- You should use Timer0 Interrupts for note generation and configure Timer0 to work in 8-bits mode.
- You should configure Timer1 to work in 16-bits mode.
- It is beneficial to avoid subroutine calls in Timer ISRs. You may use counters or state markers and you can do the relevant operations in the main routine.
- Please be careful about the seven segment display subroutine. It can be implemented either by using Timer1 interrupts or inside the round-robin loop. It should display the values in a smooth manner without flicker. You should optimize the update rate.

- You should use rightmost and leftmost display of the seven segment display. The middle ones should not be on.
- Please write your time calculations to your .c file as a comment.
- When you are writing your code, you can use PIC18F8722 datasheet and lecture notes. They contain almost all required information.

5 Hints

5.1 How to Create Random Notes?

For the proper playing of the game, we have to create notes randomly on one of the RA0-RA4. For this purpose you will use the value of Timer1 at the time of RC0 is pressed in the beginning of the game. You should Timer1 in 16-bit mode. The following algorithm is a pseudo-random note generation algorithm. The algorithm is explained with 8 bits for simplicity:

1. When RC0 button is pressed at the beginning of the game, read the value of Timer1 once and save its value. Let's assume the value is 01110010.
2. Take the rightmost 3-bits (010).
3. Apply modulo operation to the rightmost 3-bits. Resulting value is the place where the random notes will be generated.(010 mod 5 = 2 $\longrightarrow$ generate the note at RA2)
4. Rotate right the value by N. N is 1, 3, 5 for the Level-1, Level-2 and Level-3 respectively.(Rotate is basically a shift operation with preserving the leftmost bit in the rightmost position)
   - If the level is 1, then N is 1, and the new value after the rotate is 00111001.
   - If the level is 2, then N is 3, and the new value after the rotate is 01001110.
   - If the level is 3, then N is 5, and the new value after the rotate is 10010011.
5. Every time when you need to generate a random note, repeat the steps 2, 3 and 4.

5.2 Seven Segment Display

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 6. 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.

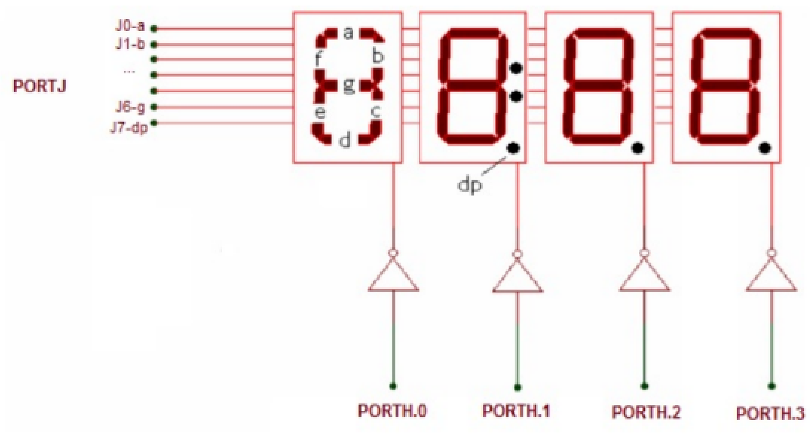

Figure 6: Connections for the seven segment displays in the MCDEV board.

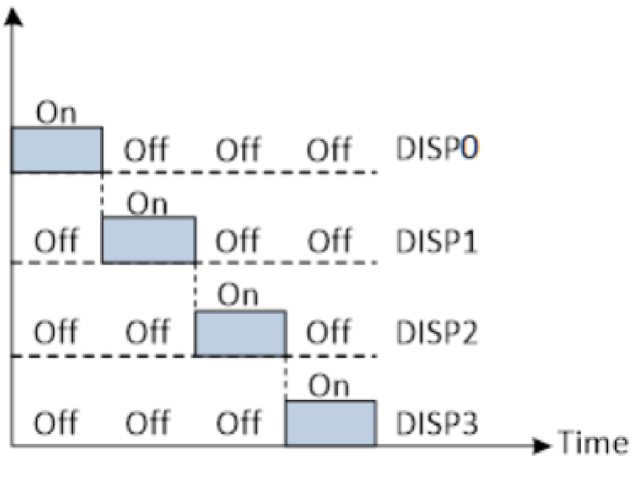

Figure 7: Graphical illustration to show characters seven segment displays simultaneously.