Middle East Technical University Department of Computer Engineering

CENG 336

Introduction to Embedded Systems Development
THE2

Due: 27th of April 2025, 23:55
Submission: via **ODTUClass**

1 Introduction

This group assignment will familiarize you with the basic interrupt operations and seven-segment display usage using PIC18F8722 microprocessor and the provided boards. You will implement a simple game similar to the "Hungry, Hungry Hippos". In this specific case, a single "Hippo" will grab the prize and grow in size. The initial state of the game can be seen in Figure 1.

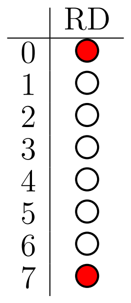

Figure 1: Initial State of the game board.

The game will be held over the LED strip of PORTD. The prize will be denoted as a single LED which is always at the top (the least significant bit of PORTD). The prize will blink continuously to distinguish it from the hippo. The hippo will be at the most significant bit of PORTD. Initially, the hippo will have a size of one.

The user will try to make the hippo reach to the prize. Interrupt-driven user input will be used to move the hippo one square up. Gravity will act over the hippo to pull the hippo down and make the game challenging. The user will try to beat the gravity to reach the prize by continuously pressing and releasing the input button.

As the hippo eats the prizes, it will grow in size. Every time the hippo eats a prize, the game will soft reset, meaning that the hippo will teleport to the bottom of the LED strip, and its size will increase by one. For example, the hippo is initially a single LED. After eating the first prize, it will be denoted with two LEDs (PORTD_6 and PORTD_7). Before the soft reset, all the PORTD LEDs will blink for some amount of time.

When the hippo has reached the size of 6, the game will hard reset. This means the hippo's size will be reset to one, and similar to the soft reset, it will teleport back to RB7. So you should never see a hippo represented by 6 LEDs.

Finally, the game will have a score element as well. When the user reaches the prize faster, they will get a higher score. The accumulated score will be shown in the 7-segment displays. That is the gist of the game. More detailed specifications are provided in the upcoming sections.

2 Specifications

2.1 Configuration

S-1 The assignment will be written for **PIC18F8722** processor.

S-2 The processor's clock speed will be **40 MHz**.

S-3 The assignment will be written in C.

2.2 Inputs

S-4 The hippo will be moved forward via PORTB_0:

S-5 PORTB_0

S-6 With a single press/release cycle the hippo will advance only one square. It is up to the implementor where this action will happen (press or release).

2.3 Outputs

S-7 PORTD will be used as output. These LEDs will define the game area.

S-8 PORTH and PORTJ will be used to drive the 7-segment display. (Please see Section 3).

2.4 Interrupts

S-9 As stated above, PORTB_0 will be interrupt driven. This means INT0 will be used.

S-10 Additionally, **TIMER0** interrupt will be utilized.

- TIMER0 interrupt will be used to "blink" the prize (its LEDs will be on and off). This will occur every **500ms**.
- Additionally, this timer will be utilized for gravity. The hippo will go "down" by a single LED every **350ms**. This movement is in addition to the user's input (via PORTB_0, which moves the hippo toward the prize).
- The timer is also used for degrading the score of each round. The maximum score a user can get in each round is 100. **Every second**, the score will degrade by 10 points. The score cannot go below zero. So if the hippo eats the prize after 10 seconds in a given round, the user will get zero points from that round.

S-11 Only a single timer (**TIMER0**) must be used.

S-12 This timer must trigger an interrupt every 5 ms, $\pm 100\mu s$.

2.5 Initialization

S-13 All LEDs should be cleared.

S-14 Interrupt and I-O-related chip states should be set.

S-15 7-segment display should show zeroes.

S-16 The system should wait **one second**, enable interrupts, and start the game loop.

2.6 Game Loop

S-17 The game will show the hippo (size one initially), and the prize to the user.

S-18 Starting from the on state, the prize will blink according to the specification above.

S-19 As the user presses the button, the hippo will go up by one square.

S-20 The gravity will pull the hippo down by one square according to the specification above.

S-21 The hippo should never go out of bounds due to gravity. Meaning, that when the user stops pressing the input button, it should fall and stay at the bottom of the screen (If the hippo has a size of 3, you should see 3 LEDs at the bottom).

S-22 The user will try to beat the gravity by repeatedly pressing/releasing the input button.

S-23 When the hippo reaches the prize, the current round score will be added to the total score. Moreover, the game will soft reset.

S-24 When the soft reset occurs, all the LEDs on PORTD will blink:

- Total blink duration is 2 seconds.
- All LEDs should be lit initially.
- On/Off duration is 400 milliseconds. This means all LEDs will be on 3 cycles and off 2 cycles (On-Off-On-Off-On)
- After two seconds, the hippo's size will be increased by one, and the hippo will be positioned at the bottom of the screen.
- A new prize will be rendered on top of the screen.

S-25 During this gameplay loop, the total score will be rendered on the 7-segment displays.

S-26 When the hippo reaches size 6, it should "overflow" and be shown as a hippo of size 1. This is called a hard reset. This means there should not be a size 6 hippo throughout the lifetime of the program.

S-27 When the program hard resets, **the total score should not be reset**. It should continue accumulating.

S-28 The game will continue looping forever in this fashion.

3 7-Segment Display

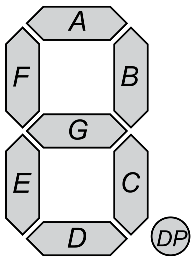

Figure 2: 7-segment display segment naming.

There are four common cathode 7-Segment displays mounted on the development board. PORTH and PORTJ are hooked to this 7-segment display. Specifications are as follows:

- PORTH_0 is connected to D_3
- PORTH_1 is connected to D_2
- PORTH_2 is connected to D_1
- PORTH_3 is connected to D_0
- $D0$ is the rightmost 7-segment display. (Please check your board. This representation assumes the I-O section of the board is towards up).
- Given these ports above, any data in PORTJ will be sent to 7-segment displays where their enable bit is one. For example, if only D_0's bit is enabled (PORTH_3 is one), only D_0 will receive the PORTJ's value.
- PORTJ represents the segments on the display:
  - PORTJ_0 is connected to A
  - PORTJ_1 is connected to B
  - PORTJ_2 is connected to C
  - PORTJ_3 is connected to D
  - PORTJ_4 is connected to E
  - PORTJ_5 is connected to F
  - PORTJ_6 is connected to G
  - PORTJ_7 is connected to DP
- We will not use DP portion of the 7-segment display.
- You can find the diagram on ODTUClass (Page 2).
- In order to use multiple 7-segment displays properly, you need to continuously write the values to each event segment display concurrently.

- **The 7-segment displays do not retain signals**. You need to send a signal to a 7-segment display, wait for some time, send a signal to another 7-segment display, and so on. If you wait too long, you will see flickering (displays will look like they are blinking). If you wait too short, the signal of one 7-segment display will smear to another 7-segment display (the 7-segment display will seem to show two numbers simultaneously). You can start with 5ms of wait time. But it will depend on your application.

4 Example Run

In this section, an example run for the program is provided. Table 1 shows the initialization and a simple gameplay of two rounds. The table timings are somewhat arbitrary since the user may not provide millisecond-correct inputs.

Table 1: Example runtime of the program. The least significant bits are the top-most LEDs on the diagram

Line No	RD	7-SD	Duration (ms)	Description
1	● ○ ○ ○ ○ ○ ○ ●	0000	0ms	Initial State.
2	● ○ ○ ○ ○ ○ ○ ●	0000	+1000ms	The game is started, and the hippo and prize are shown.
3	○ ○ ○ ○ ○ ○ ○ ●	0000	+500ms	The prize is off due to blinking. The user did not induce any actions. If user reaches to the prize in the next 500ms, they can receive the full score (100 pts).
4	○ ○ ○ ○ ○ ○ ● ○	0000	+200ms	The user pressed and released the input button (PORTB_5), and the hippo moved forward.
5	○ ○ ○ ○ ○ ○ ○ ●	0000	+50ms	750 ms have passed since the game started. Three gravity ticks have occurred during this time. The last one (happens exactly this moment) pulled the hippo down.
			+1000ms	Assume the user induces input 11 times during this period (4 gravity actions occurred, so the hippo went up by 7 units).
6	● ● ● ● ● ● ● ●	0090	+0ms	The hippo reaches the prize. The soft reset sequence is started. The score is updated. 1750 ms has passed since the round start so $(100 - 10 = 90)$ is added to the total score. All LEDs started blinking (these are initially on).

Table 2: Continuation of the Table 1

7	○ ○ ○ ○ ○ ○ ○ ○	0090	+400ms	`The soft reset continues.  LEDs are off.`
8	● ● ● ● ● ● ● ●	0090	+400ms	`The soft reset continues.  LEDs are on again.`
9	○ ○ ○ ○ ○ ○ ○ ○	0090	+400ms	`LEDs are off again.`
10	● ● ● ● ● ● ● ●	0090	+400ms	`Final ‘‘on’’ cycle of the soft reset.`
11	● ○ ○ ○ ○ ○ ● ●	0090	+400ms	`The new round starts.  The hippo is the size of two since it has eaten the prize.`

5 FAQ

1. **Q: Should 7-segment display code be derived by interrupt or in a round-robin fashion? How should we drive multiple 7-segment displays simultaneously?**

   A: This is up to you. It can be done in both ways. It should be noted that the displayed numbers should not flicker. 7-segment displays should be driven in a round-robin fashion (via either interrupt or in the main loop). Thus, you should send data to one 7-segment display and wait. Then send data to the other 7-segment displays, etc.

2. **Q: Should we utilize all 7-segment displays on the board? What if the score overflows (in terms of digits)?**

   A: Yes. Initially, the display array should render all zeroes. Overflow condition (score exceeding 9999) will not be checked.

3. **Q: When should the action of the button be induced? It should be a rising edge or a falling edge (i.e., press or release)? What would happen if the user holds the RB5?**

   A: Since the grading will happen manually, you can implement it either way. This means that you can set INTEDG0 bit as either one or zero.

4. **Q: Our boards have problems with PORTD. Should we change ports if applicable?**

   A: Yes you can change ports for display. However, due to hardware limitations, you cannot change the PORTH and PORTJ. If these ports have malfunctions, please exchange your board with another board via Erol Öztaş. The boards were tested at the start of the semester (with a similar program that you use in the-1.5), so this should not be an issue. You can change PORTB_0 with another interrupt-driven line (PORTB_0 - PORTB_3). **You cannot change the input with another port since only these bits are interrupt-driven**.

6 Grading

The grading will be done manually. Your submission will be compiled, uploaded to a board, and tested by hand. The grading criteria are given below:

- Blinking of the prize.
- Proper player controls.
- Proper act of gravity.
- Proper score calculation.
- No out-of-bounds movement.
- Proper update of the 7-segment display.
- Non-flickering 7-segment display.
- Correct blink sequences.

These conditions may be checked on multiple hardware resets. The exact rubric has not been finalized and thus has not been provided. All of the conditions above will be tested and graded in isolation to the extent possible.

Additionally, your code will be checked in the simulator manually. In this case, only the timer will be checked to make sure it is in range of the specification given above.

7 Regulations

1. The assignment will be done **in groups**.
2. Please track the forum for discussions and questions about this assignment. Other students may have already asked your question, which may have been answered already.
3. If your issue is related to *your code* please either e-mail me `yalciner@ceng.metu.edu.tr` or come to my office `A210`.
4. Please consult your lecture notes and datasheets before asking any questions.
5. **Submission:** Please upload the filled student pack as a single *.zip file. It should be in a compilable state.
6. **Late Policy:** You can extend the deadline by **3 days maximum**. If you extend the deadline by n days, you will receive a grade penalty of $5n^2$.
7. **Board Usage:** Since a subjective evaluation will be conducted, your solution must work on the boards to get the full grade.
8. **Using Large Language Models (LLMs) and other AI tools:** We note that AI can be a useful learning resource, but if it is used for generating code that you are supposed to write, it can have a detrimental effect on your learning. Therefore, **using code from other resources, whether it is generated by humans or AI, is considered cheating**.
9. There are plenty of resources available to you in ODTUClass. To refer to the instructions and specifications of PIC18F8722, you should use the PIC18F8722 Datasheet. Since you will be using the MPLAB X IDE simulation environment for this assignment, the Recitation 1 documents can also be a great setup guide. If you set up a local environment, you should use MPLAB X IDE version 5.45 and XC8 Compiler version 2.30; both are available in the downloads archive. You can use the department labs if you cannot set up a local environment.