



Middle East Technical University



Department of Computer Engineering

CENG 336

Introduction to Embedded Systems Development THE2

Due: 22th of April 2024, 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. To accomplish this, you are going to implement a simple Tetris-like game. This game will have three shapes (Figure 1).

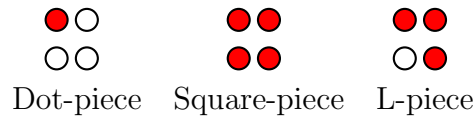


Figure 1: Shapes of the Tetris game

The game will be held over the LED area of ports PORTC-F. An illustration of this is given in Figure 2. The user will provide commands to move a Tetris piece and try to submit it on this board in an appropriate position. When a piece is submitted on the board, it will stay there until the system is reset. Details of the rules and input specifications will be given below.

Finally, the game logic should display how many blocks are occupied in the game area. This will be done using the 7-segment display. The game will end when all blocks in the game area are occupied.

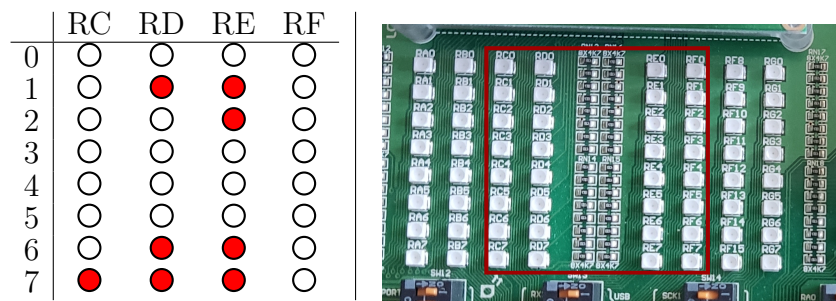


Figure 2: LED board example and area on the actual PICKit.

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 Tetris pieces will be submitted to this board via user inputs from $PORTA$:

- $PORTA_0$ move the piece *right*
- $PORTA_1$ move the piece *up*
- $PORTA_2$ move the piece *down*
- $PORTA_3$ move the piece *left*

S-5 $PORTA$ will be **polled**.

S-6 Tetris pieces will be rotated via $PORTB_5$:

S-7 Tetris pieces will be submitted via $PORTB_6$

S-8 Both $PORTB_5$ and $PORTB_6$ will be triggered by interrupts.

2.3 Outputs

S-9 $PORTC-F$ will be used as outputs. These LEDs define the game area.

S-10 $PORTH$ and $PORTJ$ will be used to derive the 7-segment display. (Please see Section 3).

2.4 Interrupts

S-11 As stated above, $PORTB_5$ and $PORTB_6$ will be interrupt driven.

S-12 Additionally, **TIMER0** interrupt will be utilized.

- **TIMER0** interrupt will be used to “blink” the current moving tetris piece (Its LEDs will be on and off).
- Additionally, this timer will be utilized for Tetris piece movement. The current Tetris piece will go “down” gradually. This movement is in addition to the user's input (via $PORTA$).

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 system will provide a Tetris piece to the user.

- Until the user submits this piece, this piece will be “alive”.
- The user can interact with this piece using input buttons.
- This piece will gradually fall every **2 second**. This means the live piece will vertically drop one square (LED) every 2 seconds.
- This piece will be blinking every **250ms**. **The first 250ms the piece will be on.** Next 250ms, it will be off, and so on.
- User may rotate this piece at any time ($PORTB_5$). The piece will be rotated in **clock-wise order**.

The rotation makes sense only for the L-piece. For all other shapes, you can ignore the rotation. The rotation order of the L-piece is given in Figure 3.

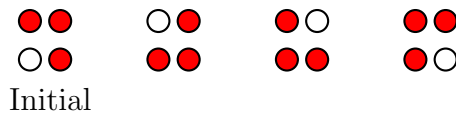


Figure 3: Rotation order of the L-piece.

S-18 When the user is satisfied with the current location of the “alive” piece, it may press the submit button ($PORTB_6$).

S-19 Submission can only occur if the current location of the piece is empty. Suppose any previously submitted pieces overlap with this piece, **submission should not happen**.

S-20 This means that the live piece will continue accepting inputs, blinking, etc.

S-21 If the submission is successful:

- The corresponding LEDs will be lit through the program’s runtime.
- 7-segment display will be updated with the new value of the occupied location count on the game board.

S-22 User may provide unnecessary inputs that may force the live piece to move out-of-bounds of the game board. **The program should not allow this to happen.** Meaning:

- If the piece moves to the far right or far left, and the user provides an out-of-bounds input sequence, the piece should remain on the far right and far left location on the game board.
- The same is true for up and down movement as well as timer-induced “fall” movement.

S-23 Alive pieces are provided to the user in a predetermined fashion. The very first piece will be **Dot-piece**, the second one will be **Square-piece**, and the third piece will be **L-piece**. After that, the sequence will fall back to the Dot-piece.

S-24 Alive pieces spawn on the “top left” corner of the game board ($PORTC_0$). The large piece’s top left corner should be aligned with $PORTC_0$.

S-25 The Game will end (the system will reset when) when the entire game board is covered with Tetris pieces. The user may lock the game when no appropriate place exists for the current live piece.

3 7-Segment Display

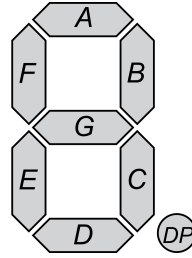


Figure 4: 7-segment display segment naming.

There are three 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
- D_0 is the rightmost 7-segment display. (Please check your board. This representation assumes I-O boards of the boards 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

- $\text{PORT}J_1$ is connected to B
 - $\text{PORT}J_2$ is connected to C
 - $\text{PORT}J_3$ is connected to D
 - $\text{PORT}J_4$ is connected to E
 - $\text{PORT}J_5$ is connected to F
 - $\text{PORT}J_6$ is connected to G
 - $\text{PORT}J_7$ is connected to DP
- We will not use DP portion of the 7-segment display.
 - You can find the diagram on [ODTUClass](#) (Page 2).

4 Example Runtimes

In this section, we will provide two example runtimes for the programs. Table 1 shows the initialization and first piece falling, and the user submits the Dot-piece to the bottom left position. The second example (Table 2) is from the middle of a game. The user tries to submit a piece but can not due to no space available. 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


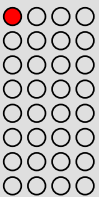
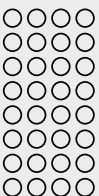
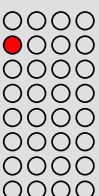
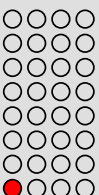
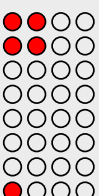
Line No	RC/RD/RE/RF	7-SD	Duration (ms)	Description
1		0000	0ms	Initial State.
2		0000	+1000ms	The system is started, and the very first piece is coming down.
3		0000	+250ms	Blinking.
4		0000	+1500ms	The piece fell due to ‘‘fall’’ logic.
...				Assume some time has passed. No user input is provided.
5		0001	+Xms	Dot-piece slowly fell to the bottom. While doing so, it was blinking. The user pressed $PORTB_6$ (Submit). That portion on the board was empty, so the piece was written on the game board.
6		0001	+10ms	Immediately after the next piece is spawned, it is the Square-piece. Notice all the pieces are spawned top-left.

Table 2: Advanced example. During a game user rotates a piece and tries to submit but fails.

Line No	RC/RD/RE/RF	7-SD	Duration (ms)	Description
	...			An ongoing game...
1		0009	Xms	Current state of the game. Assume the user just submitted a piece.
2		0009	+10ms	Immediately after, an L-piece is spawned (Assuming the next piece was an L-piece).
3		0009	+50ms	L-piece is rotated twice.(by pressing PORTB5)
4		0009	+100ms	L-piece is moved to ‘‘PORTE ₂ PORTE ₃ PORTF ₂ ’’ position. It is currently overlapping with the game board at PORTE ₂ .
5		0009	+50ms	Assuming the user pressed the button PORTB6. Nothing happens due to the collision between the live piece and the game board.
	...			The game continues...

5 FAQ

1. **Q: What will happen during rendering a large piece (multiple-LEDs) and an interrupt occurs (Either due to fall-down or rotation)?**

A: This is indeed a concern and should be handled by you. As a hint, you should collapse both “rotation” and “fall-down” logic into single variables and handle rendering/rotating inside the main loop.

2. **Q: Should 7-segment display code be derived by interrupt or in a round-robin fashion? How should we derive 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.

3. **Q: Should we utilize all 7-segment displays on the board?**

A: All unused displays should display “zero” at all times. There are $8 \times 4 = 32$ LEDs on the game board, so two seven-segment displays should be enough. **Rightmost two seven-segment displays should be utilized for displaying numbers.**

4. **Q: Can live piece pass through the solid pieces (game board)?**

A: Yes, the live piece can move through the solid pieces. Only it can not be submitted if it collides with the solid pieces.

5. **Q: What happens if the live piece is inside solid pieces (either partially or fully)? Should the collided piece blink or not?**

A: Collided portions of the piece should blink as discussed above. This means the rendering of the live piece *supersedes* the rendering of the solid pieces.

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:

- Moving pieces via PORTA.
- Proper live piece order (Dot, Square, L, and so on).
- Submission of the piece (Bot correct and incorrect states).
- No out-of-bounds movement.

- Proper update of the 7-segment display.
- Non-flickering 7-segment display.
- Rotation of the L-piece.
- Blinking of the live piece.

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 an as isolated fashion as possible.

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 be 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**. Each day you extend the deadline, **10 points** will be deducted from your grade.
7. **Board Usage:** Since a portion of the grading will be done, Your solution should work on the boards to get the full grade.
8. 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 MPLAB X IDE simulation environment for this assignment, 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.