
Middle East Technical University


Department of Computer Engineering

CENG 336

Introduction to Embedded Systems Development

THE1

Due: 20th of March 2025, 23:59

Submission: via **ODTUClass**

1 Introduction

The purpose of this individual assignment is to familiarize you with the basic I/O operations and round-robin approach on PIC18F8722 using MPLAB X IDE simulation environment. To this end, you will implement a device that is capable of displaying a number sequence using LEDs according to the user input.

The program will use an 8-bit output port connected to LEDs to display a sequence of numbers in binary format, while blinking a LED in another port. Using a round-robin approach, the program will maintain precise timing for both progress bars while handling the user input. The user will be able to pause/resume the sequence and modify the numbers in it.

1.1 Notation

PORTX denotes an input/output port of PIC, where X is the letter of the port. LATX is the latch corresponding to PORTX. This assignment will use PORTB, PORTC, PORTD and PORTE. For a given PORTX, RXi for $0 \leq i \leq 7$ denotes the ith least significant bit in PORTX. For example, RB0 is the first (smallest) bit in PORTB. A single bit in a port is termed a "pin".

2 Specifications

2.1 Configuration

S-1 The program shall be written for **PIC18F8722** running at **1 MHz instruction frequency**.[1]

S-2 PORTC and PORTD shall be used for output.

- PORTC shall display the current number in binary.

[1]Note that the instruction frequency of the boards given to you is 10 MHz. Consequently, the program will run 10 times faster on that hardware. In this assignment, we decreased the instruction frequency to reduce the computational load of the simulation.

- RD0 shall be a blinking LED.

S-3 PORTE shall be used for input.

- REi for $i \in [0, 5]$: Increment the number at index i in the sequence.
- RE7: Pause/resume the sequence (but not the blinking LED).

2.2 Initialization

S-4 The program shall initialize all variables it uses, i.e., it shall not depend on variables being zero at the start.

S-5 The program shall light up all the LEDs in PORTC and PORTD for 1000 ± 50 ms at the startup. This is called the **initialization period**.

S-5.1 The program does not need to respond to any inputs during this period, i.e., it can busy-wait. In this context, **busy-waiting** refers to a loop whose only purpose is to cause a time delay. During this loop, the program does not check for inputs and hence it's unresponsive.

S-6 After that, the program shall begin operation.

2.3 Blinking LED

S-7 Immediately after the initialization period, RD0 shall turn off for 500 ± 50 ms and then light up for 500 ± 50 ms, repeating this cycle indefinitely.

S-8 The blinking LED at RD0 shall continue even when the sequence is paused using RE7.

2.4 User Interaction

S-9 A "release" of a button is defined as a change from 1 to 0 in the corresponding pin.

S-9.1 Each pin is 0 by default, becomes 1 when pressed, and becomes 0 when released.

S-9.2 The program shall not busy-wait for the button release (1 to 0) after the button press (0 to 1). While the button is held (corresponding pin is 1), the program shall operate normally as if nothing happened. Only when a held button is released (1 to 0), a click is registered.

S-10 The program shall **NOT** implement button debouncing. Any change from 1 to 0 is a release, regardless of the duration or previous changes.

2.5 Number Sequence

S-11 The number sequence is a list of six numbers.

S-12 Each number is between 0 and 9, inclusive.

S-13 The initial values in the number sequence are the digits of your 6-digit student ID, starting from the most significant digit.

S-14 PORTC shall display the current number in the sequence in binary format.

S-15 The sequence shall begin immediately after the initialization period, i.e., PORTC shall display the first number when 1000 ± 50 ms duration ends.

S-16 The sequence shall progress when RD0 changes state (every 500 ± 50 ms).

S-17 A button release in RE7 shall pause/resume the progression of the number sequence without affecting the blinking LED.

S-18 After the last nubmer in the sequence, the program shall wrap around to the first number in the sequence.

S-19 A button release in REi for $i \in [0, 5]$ shall increment the number at index i in the sequence. If the incremented value exceeds 9, it shall be set to 0.

S-20 If the currently displayed number is incremented, the behavior is unspecified, i.e., it may update the number immediately or keep displaying the old value. This case will not be tested.

2.6 Implementation Details

S-21 The program shall use busy-waiting for the 1-second initialization period.

S-22 The program shall use a round-robin approach for the main operation and input handling.

S-23 Timers and interrupts are not allowed.

S-24 The program shall only write to the output ports when they need to change, i.e., once every 500 ± 50 ms. The program shall **NOT** perform unnecessary writes to the output ports in each iteration of the main loop. This is required for testing.

3 Example Runs

In this section, example runs of the program are given in tables.

- ○ represents that the LED is off, i.e., the corresponding pin is 0.
- ● represents that the LED is lit, i.e., the corresponding pin is 1.
- 8 LEDs denoted by ○○○○○○○○ represent an 8-bit output port. The rightmost LED is the least significant bit. For example, ○○○○○○●● represents the decimal value of 3 on the corresponding output port.

The 6-digit student ID is assumed to be 123456. The duration represents how long the given LED configuration is maintained before advancing to the next row. Comments and inputs are provided as special rows in each table.

Table 1: Example run with no input. The student ID (initial number sequence) is assumed to be 123456.

PORTC	PORTD	Duration (ms)
●●●●●●●●	●●●●●●●●	1000
Initialization period ends, main operation begins.		
○○○○○○○●	○○○○○○○○	500
○○○○○○●○	○○○○○○○●	500
○○○○○○●●	○○○○○○○○	500
○○○○○●○○	○○○○○○○●	500
○○○○○●○●	○○○○○○○○	500
○○○○○●●○	○○○○○○○●	500
Last 6 steps repeat indefinitely.		

Table 2: Pausing example. The student ID (initial number sequence) is assumed to be 123456.

PORTC	PORTD	Duration (ms)
●●●●●●●●	●●●●●●●●	1000
Initialization period ends, main operation begins.		
○○○○○○○●	○○○○○○○○	500
○○○○○○●○	○○○○○○○●	500
○○○○○○●●	○○○○○○○○	200
A button release occurs at RE7 pin.		
○○○○○○●●	○○○○○○○○	300
○○○○○○●●	○○○○○○○●	500
○○○○○○●●	○○○○○○○○	500
○○○○○○●●	○○○○○○○●	300
A button release occurs at RE7 pin.		
○○○○○○●●	○○○○○○○●	200
○○○○○●○○	○○○○○○○○	500
○○○○○●○●	○○○○○○○●	500
○○○○○●●○	○○○○○○○○	500
...		

Table 3: Increment example. The student ID (initial number sequence) is assumed to be 123456.

PORTC	PORTD	Duration (ms)
●●●●●●●●	●●●●●●●●	1000
Initialization period ends, main operation begins.		
○○○○○○○●	○○○○○○○○	500
RE2 is released.		
○○○○○○●○	○○○○○○○●	500
○○○○○●○○	○○○○○○○○	500
○○○○○●○○	○○○○○○○●	500
RE5 is released 3 times.		
○○○○○●○●	○○○○○○○○	500
○○○○●○○●	○○○○○○○●	500
Sequence starts over. RE5 is released again.		
○○○○○○○●	○○○○○○○○	500
○○○○○○●○	○○○○○○○●	500
○○○○○●○○	○○○○○○○○	500
○○○○○●○○	○○○○○○○●	500
○○○○○●○●	○○○○○○○○	500
○○○○○○○○	○○○○○○○●	500
Note that the last number wrapped around to 0. Last 6 steps repeat indefinitely.		

3.1 Grading

Your submission will be tested using black-box tests. However, for technical reasons, your program needs to abide by the following rules for correct testing:

1. **DO NOT** perform unnecessary writes to the output ports in each iteration of the main loop. Tests use the `watch` command in `mdb` to track changes in ports. If you keep rewriting data to the output ports in every iteration, the debugger will be overwhelmed and the test will fail.

2. When the simulator first runs, all variables will be set to 0. You program **MUST NOT** depend on this behavior. The tester works by performing a hardware reset after each test case. After a hardware reset, the variables will **NOT** be set to 0. If you assume that all variables are set to 0 at the start, it's likely that your program will exhibit peculiar behavior after the first test case.

3. Make sure that there are **no non-ASCII characters** (e.g. special Turkish characters) in your source code. Otherwise, you may get `UnicodeDecodeError` while running the tester. Even if you don't encounter any errors, avoid non-ASCII characters because they may cause issues during grading.

Please note that not abiding by these rules may result in **0 grade** due to complete testing failure, or a significant grade reduction.

The tentative grading rubric is given in Table 4. → denotes that the grading item is a subitem and depends on its parent, the closest previous grading item. If the program fails the parent grading item, all of its subitems are automatically failed. Please keep in mind that this grading rubric does not include the late submission penalty (if applicable) and other external penalties that may incur due to cheating or using forbidden features (e.g. timers and interrupts).

Table 4: Tentative Grading Rubric.

Criterion	Points
Initialization period (2.2)	5.0
Blinking LED test without input	5.0
Number sequence test without input	7.5
→ Wrap around after the last number	7.5
Pause sequence on RE7	7.5
→ Resume sequence on RE7	7.5
→ Pause again on RE7	5.0
→ Resume again on RE7	5.0
Increment number on RE0 to RE5	20.0
→ Reset to 0 after exceeding 9	10.0
Continue normally while a button is held down	20.0

In order to test your submission, please **go over the items in the grading rubric and confirm that each item works correctly in the simulator**. Inspect the PORT values in the debugger by putting breakpoints and use the stopwatch to verify the timing.

4 Regulations

1. This is not a group assignment, you must solve it **individually**.
2. Any clarifications and revisions to the assignment will be posted to ODTUClass. You are expected to follow the announcements so that you are up to date.
3. Ask your questions in the discussion forum dedicated for this assignment in ODTUClass. Unless you have *specific* question about *your code*, use the forum. If you think that your question is too specific to ask on the forum, you can ask your questions via email to İlker: `ilker@ceng.metu.edu.tr`
4. Please consult your lecture notes and datasheets first before asking any questions.
5. **Submission:** Upload a single `*.s` file to the ODTUClass assignment. The file name does not matter as long as the extension is correct.
6. **Late Policy:** You can extend the deadline by 3 days at maximum. If you extend the deadline by n days, you will receive a grade penalty of $5n^2$.
7. Please keep in mind that this is an individual assignment. You are not allowed to reveal your solution to other group members in any way, shape, or form, even partially.
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. Please **use the provided starting project**. Otherwise, you may encounter problems and lose points due to incorrect configuration.

5 Hints

1. There are plenty of resources available to you in ODTUClass. To refer to 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 set up 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. If you cannot set up a local environment, you can use the department labs.
2. MPLAB IDE X v5.45 direct download links:
   - Linux
   - Windows
   - macOS
3. XC8 Compiler v2.30 direct download links:
   - Linux

- Windows
- macOS

4. **Stopwatch** tool of the simulator can be used to measure time spent by a code segment, by adding **breakpoints** to the start and end of that segment. You can find this window from **Window → Debugging** menu. It is important to configure your instruction frequency to 1 MHz from **Project properties → Simulator → Instruction Frequency** before starting the simulator for the time values to be correct. (1 MHz is the default but double-check to make sure.) **Project Properties** window can be reached by right clicking the project name in **Projects** panel. You can refer to Recitation 1 documents for detailed explanations and screenshots.

5. You can also see the states of your ports, variables and pins by using the Logical Analyzer, SFR and Variables windows. They can be found under the **Window** menu in **Simulator, Debugging, Target Memory Views** sections respectively.

5.1 Troubleshooting

1. **pic-as is not installed:** pic-as should be installed automatically alongside XC8 Compiler v2.30. If it's not recognized, please check if the `pic-as` folder is present under the XC8 installation directory. If this folder is present, you can add pic-as manually using the instructions in Figure 1. The default XC8 installation directories are as follows:

   - Linux: `/opt/microchip/xc8/v2.30/`
   - Windows: `C:\Program Files\Microchip\xc8\v2.30\`

2. **Cannot fire stimulus:** When you fire an event in the stimulus window, you should see a notification like "Asynchronous stimulus RB0 fired" at the bottom of the window. If it doesn't appear and the stimulus doesn't happen (note that the stimulus becomes effective only after you step/continue the debugger), this issue might be caused by **non-ASCII characters in your source code** or the encoding. Your code must NOT contain any non-ASCII characters. If the issue persists after removing all non-ASCII characters, try replacing your `.s` file with either `main.s` provided in the starting project to ensure that the file has the right encoding.

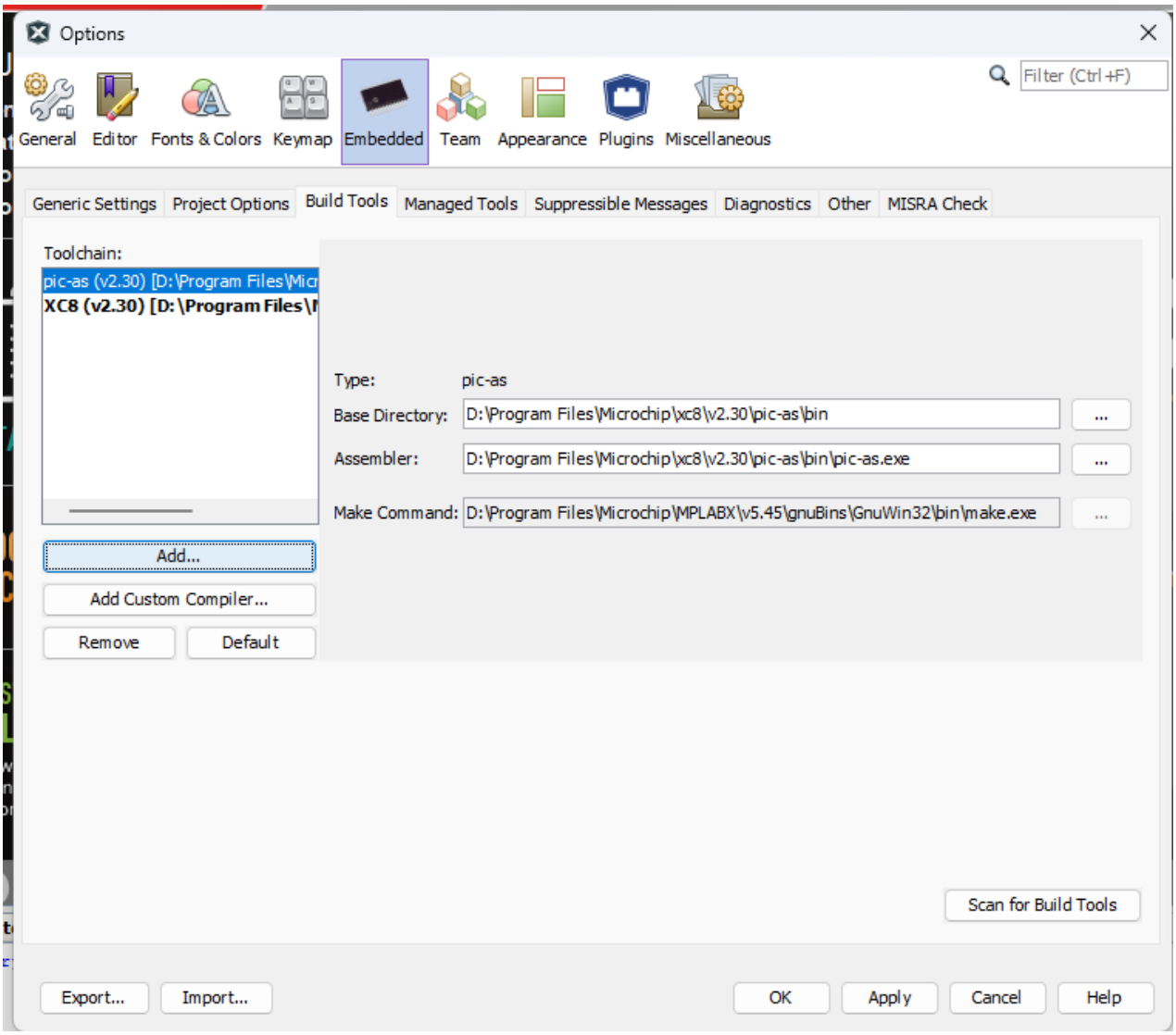

Figure 1: Adding pic-as manually.