Western Engineering

ECE 9047 Sensor Networks & Embedded Systems

Laboratory 2

Course Instructor: Dr. John McLeod

Student Number: 251343797 .

Student Name: Yuanpeng Wang .

Table of Contents

1

Description of the problem

Create a three-digit decimal counter that accurately counts 1 s intervals. The count should be displayed as a decimal number on the 7-segment display. An interrupt-enabled push button should be used to reverse the direction of the counter. The count should roll over to zero after reaching 999 (if counting up), or vice versa if counting down.

Your program should run in a continuous loop.

You **must** use interrupts to handle the push button input. You may continue to use interrupts with the timer to handle the 1 s counter or poll the timer for the 1 s interval. You **must** use the timer to count for 1 s.

2

The flowchart of my solution

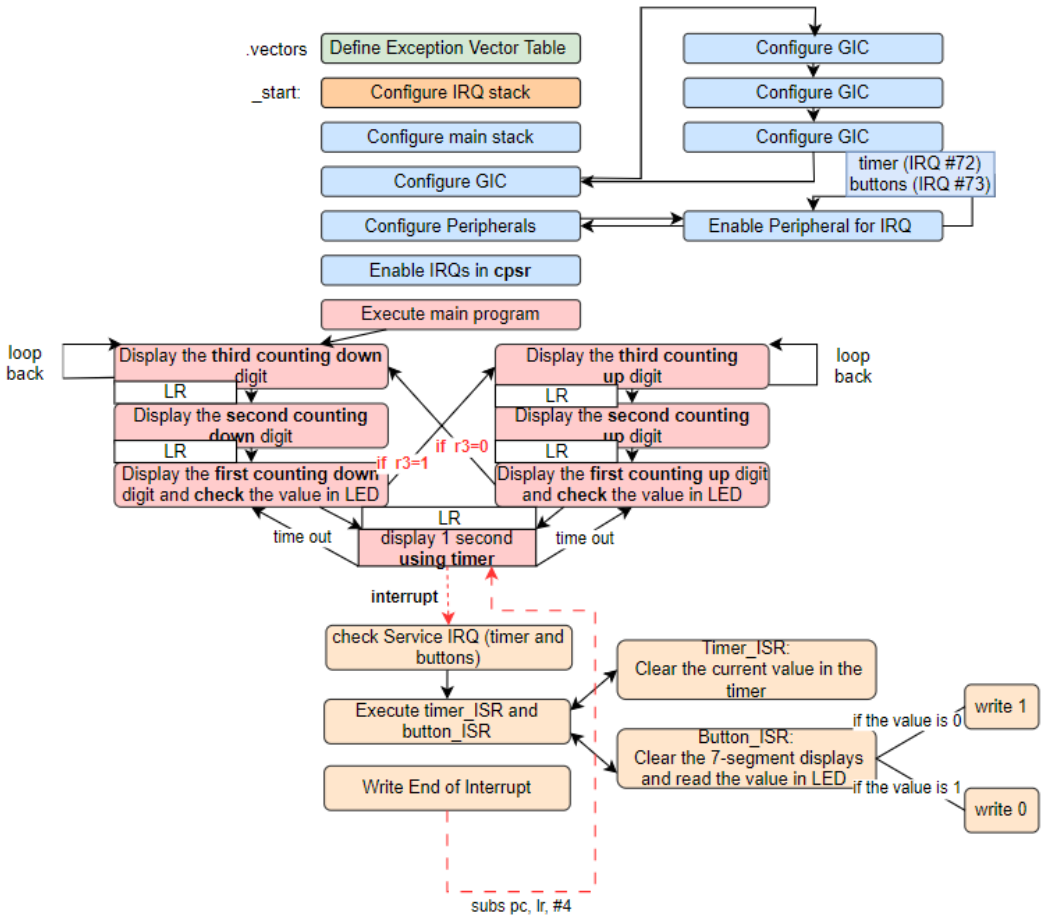

Fig. 1 The flowchart of my solution

3

Description of my approach

Description of my approach:

1. Initial Setup:

- Define the exception vector table.
- Configure IRQ stack.
- Configure the main stack.
- Configure peripherals with their IRQ number (the timer and buttons).
- Enable IRQs in cpsr.

2. Pause for 1 s:

- Use the subroutine with the link register to pause 1 second for the first digits.
- Read the status of the interval timer until it is **not equal to 2**.

3. Write to the 7-segment displays

- Separate the counting up and the counting down **process using two nested loops to display each number individually** with the link register.
- Use **the first digit** in the counting up and the counting down process to **detect the interrupts** from the push buttons.
- **Loop back** to the previous digit when the current counting up digit reaches 9 (counting down is 0) using the link register.

4. Accept interrupts from the push buttons

- Check the current value in the LED every time before displaying the first digit.
- If the current value in LED is 0. Then the main program will jump to a counting-down loop **to change the direction of the counter. (I used LED as a status direction indicator)**
- If the current value in LED is 1. Then the main program will jump to the counting-up loop **to change the direction of the counter.**

Validation:

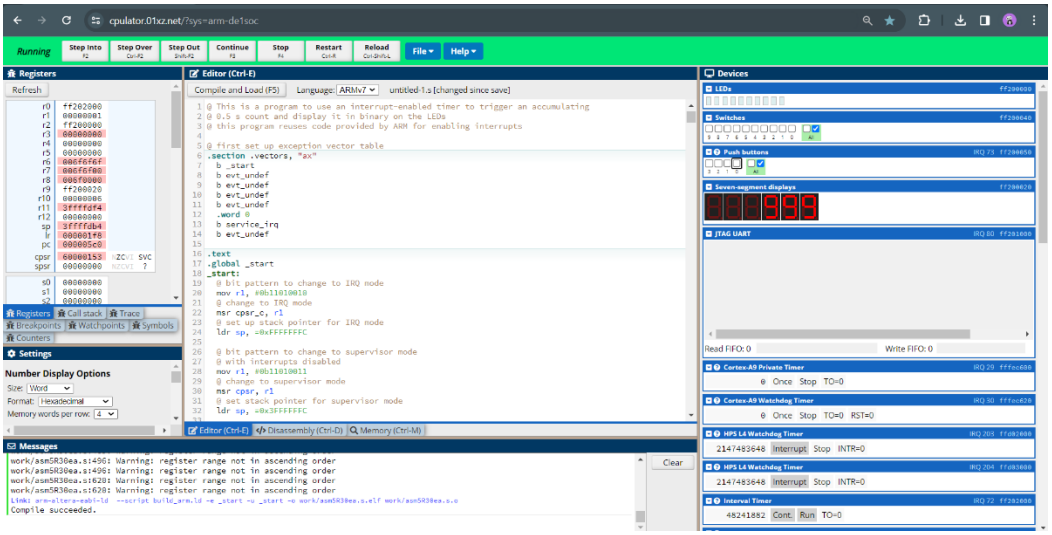

Fig. 2 Counting down

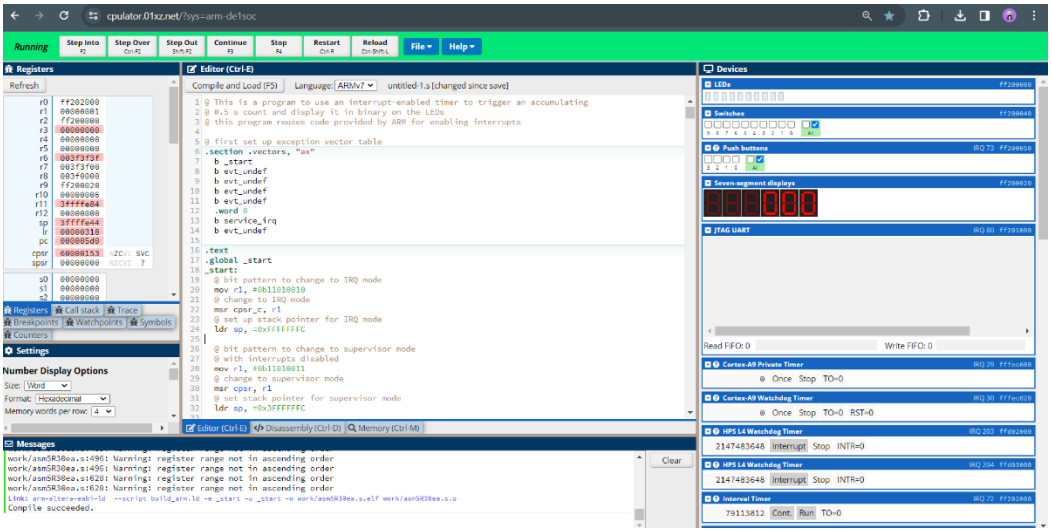

Fig. 3 Roll over to 999 after reaching to 000

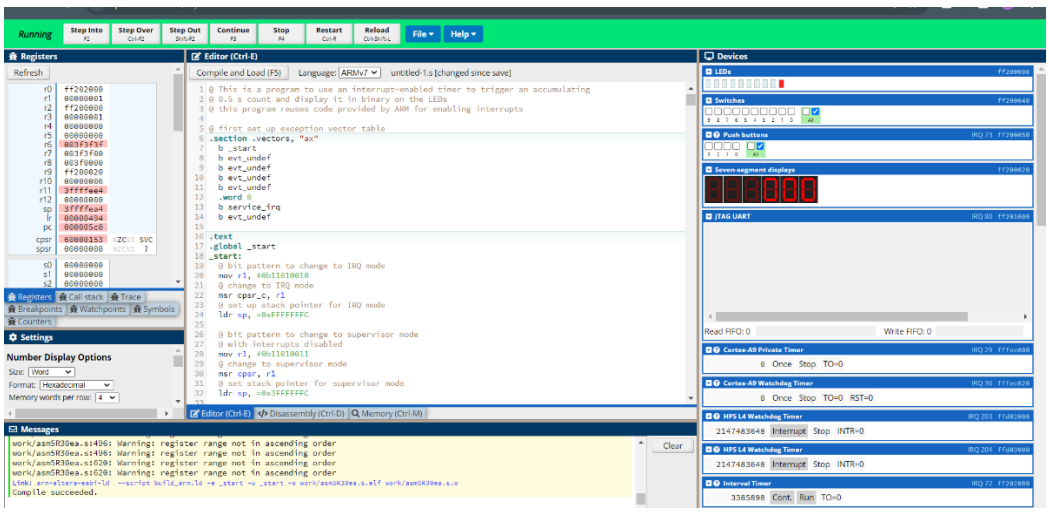

Fig. 4 Counting up

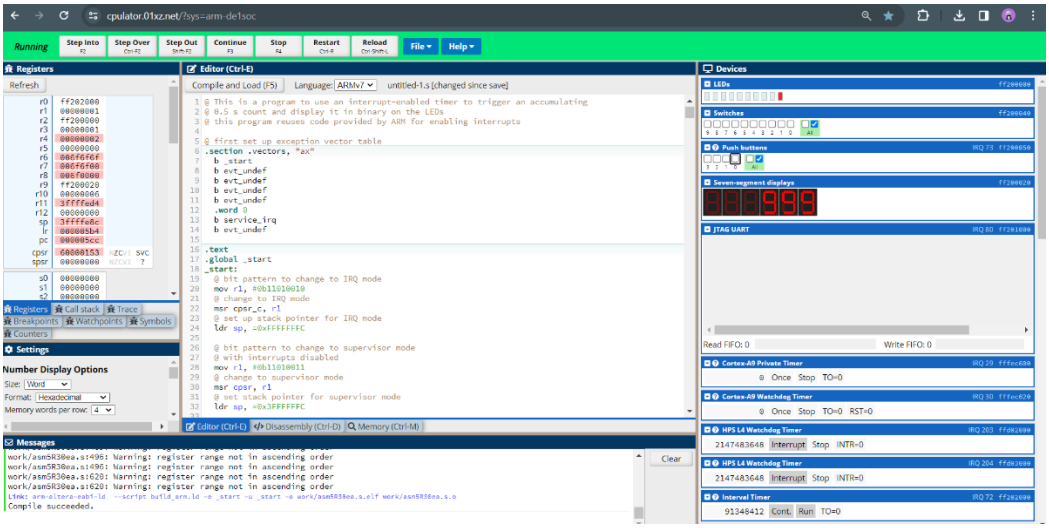

Fig. 5 Roll over to 000 after reaching to 999

4

A qualitative cost/benefit analysis of my solution

Design and Implementation:

The implementation took me 1 day to come out with the logic structure of the solution. I based on the 'Interrupt_Example' file to use a similar approach to enable IRQ services for the interval timer and the push buttons.

My solution represents my understanding of the implementation of nested loops and IRQ interrupts with peripherals.

Moreover, my solution for displaying three digits counter is very portable as it clearly separates pausing for 1 s, writing to the 7-segment displays, and accepting interrupts from the push buttons.

Maintenance and Extension:

The code structure and the use of industry-standard comments significantly enhance its maintainability. Since each section is clearly defined, and the purpose of each part of the code is well documented, it is easier for future developers to understand and modify the program.

The module design, especially in handling the display logic and interrupt service routines (ISRs), allows for easy extension. For example, integrating additional peripherals or changing the display logic can be accomplished with minimal adjustments to the existing code.

The main loop can also be customized with different functions for different purposes.