Problems

Problem 1

In this problem, you will work on fault generation and handling.

- Create a HardFaultHandlerthat will detect possible faults. The handler should properly restore stack pointer, and call reset handler aftenvards. After compiling, examine the routine to make sure there are no overheads. The exact name of this function can be seen in startup_xxx file under include directory.
- Explain the type of faults that we discussed in the class and implement a routine for each case. Observe the hardfault operation, check the context after a fault occur, and explain the behavior.
- Attach an external button to your board, and using polling method execute these hardfaults randomly. For this you can use rand( ) C library function.
- Optionally you can implement a simple toggle LED mechanism in the beginning of your code to see the reset operation.

You might need to use inline assembly for handling routine which can be used as

```
asm(tpovs r9, re');
```

in your C functions.

Problem 2

In this problem, you will work with external interrupts.

- Setup and implement an external interrupt routine with highest possible priority, and tie it to your external button. Upon pressing, your interrupt routine should execute the previously implemented hardfaults randomly.
- When interrupt happens, check out the context and explain each section.
- Your main while loop should be empty for this problem since everything will betaken careofin the interrupt handler routine.

External interrupts are used to detect edge transitions on external signal. Both rising and falling edge transitions can be configured and detected if the hardware supports it. For Cortex-MO + , external interrupts are connected to an external interrupt MUX, and each pin can be configured with the desired edge trigger mechanism.

First make sure to read Section 12 on RM0444, and understand external interrupt operation. (You can skip the parts about wakeup) Overall, to enable an external interrupt:

1. Depending on your external button connection pin and configuration (active-high or -low) enable rising or falling edge trigger from EXTI _RTSR or EXTI_ FTSR registers. Your connected pin will be the bit that you should enable (i.e. write 1). For example, if you connect the button to PC8 pin in an active-high configuration, you should write (1 8) to EXTI_RTSR.
2. Write the source port to the appropriate register EXTI_ EXTICRx. There are a total of EXTI_ EXTICRx registers, and each register is used for selecting the source port for pins. Check Table 55 on Rl','iouzv

and relevant register definition for specific configurations. For PC8 example, from Table 55 on RM0444, pin 8 can be configured to PortC by writing the first 8 bits of EXTI_EXTICR3 register.
3. Enable external interrupt from NVIC, and set priority.
4. Create external interrupt handler.

Problem 3

In this problem, you will work with multiple sources for external interrupts and priorities.

- Connect two buttons and two LEDs to your board. Enable external interrupt for both of them. Each button should toggle one LED.
- Connect one buttonin one of the following groups: {0,1}, {2,3}, {4-15} and the other in the remaimn group.
- You will use two external interrupt handlers. Make sure to check using the pending bit if the correct interrupt is received.
- Assign different priorities to your buttons, and inside the handlers create a long delay that will last for at least 5-10 seconds) o You can implement this behavior by turning on the associated LED in the beginning, then an empty for loop, then turning off the associated LED.
- Observe the preemption by trying to interrupt the current interrupt routine. (i.e. while one of the LEDs is on, press the other button to interrupt.) Explain your observations.