Practicum 4: Interrupts and DMA

General Instructions

- Use the template (provided in Moodle) to create new projects for each task.
- Ensure that all **calculations** along with **formulas** and their **relevant parameter** necessary for the tasks are documented within the report. Make sure to list all the **values** you used for the parameters. Provide an explanation if **customized parameters** and values are chosen.
- Code has to be **clearly structured** and follow **common coding guidelines:**
    - ✓ Use **meaningful variable names** that reflect their purpose or content. Avoid single-letter variable names except for loop counters (e.g., i, j, k).
    - ✓ **Add comments** to explain complex logic, algorithms, or any part of the code that might not be immediately clear to others (or yourself in the future).
    - ✓ Use **constants** (#define or const) to define constants instead of hardcoding values. This improves code readability and makes it easier to update values later.
    - ✓ Break your code into **smaller, logical functions** to improve readability, maintainability, and reusability. Each function should ideally perform a single, well-defined task.
    - ✓ **Avoid using "magic numbers"** (hardcoded numeric constants) in your code. Instead, define them as named constants with descriptive names.
- Please ensure to **submit a report** as part of this exercise. Kindly adhere to the **guidelines** outlined in the provided **template** when creating the report.
- If you encounter any issues regarding the tasks, please consult the **FAQ section** in **Moodle**. If you cannot find the answer you're looking for, feel free to post your queries in the **forum** or contact us directly via email.

Useful Tips:

- For this exercise, you can use any of the following serial monitors: putty serial client, Arduino serial monitor, HyperTerminal, serial monitor extension from VS Code.
- Check the application notes as well as the data sheet for necessary register values and formulas.

For a detailed description of these functions, please refer to the function definitions in the HAL library documentation (https://www.st.com/resource/en/user_manual/um1725-description-of-stm32f4-hal-and-lowlayer-drivers-stmicroelectronics.pdf).

Source code for the library is located inside the Drives folder (Project/Drivers/STM32H7xx_HAL_Driver/Src/*.c

Task A: Blink an LED with External Interrupts

This task is similar to the task D of practicum 1 but now you are required to use external interrupts (EXTI) on PJ12 and PJ13 to detect user inputs. You should also use the timer for the blinking cycle (as introduced in practicum 2). The EXTI line (connected to PJ12) should be used to increase the cycle time while the EXTI line (connected to PJ13) should be used to decrease the cycle time. The LED located near the 7-Segment display should be used for digital output. The initial value of the blinking cycle time should be 100 ms. The range of the blinking cycle time should be between 100 and 1000 ms. The respective interrupts either increase or decrease the current blinking cycle time by 100 ms. Pressing the buttons should update the blinking frequency with immediate effect.

Requirements:

- ✓ Minimum cycle time = **100 ms**
- ✓ Maximum cycle time = **1000 ms**
- ✓ Use **PJ7 (SEGDP)** as digital output
- ✓ Use **PJ12 (BTN1)** and **PJ13 (BTN2)** to generate external interrupts
- ✓ One step = **100 ms**

Task B: Blink an LED using Non-Blocking DMA and Timer Interrupts

In this task, we will again work on our blinking LED. However, in this task, a user interface should be implemented. You are required to control the Timer 4 with user input via UART4. The user should be allowed to start and stop the timer. At each interrupt of Timer 4, the LED of the 7-segment display must be toggled.

For user input, the following command structure must be used: #X*.

- '#' stands for the start of a command
- 'X' stands for the command ID number:
  - 1 = Start timer
  - 2 = Stop timer
- '*' marks the end of a command

(Start command example: #1*)

The UART4 interface must be configured for a non-blocking DMA receiving mode. The Timer 4 must be configured to generate an interrupt every 100 ms. Don't forget to implement some error handling. Wrong commands should be handled gracefully. Give helpful feedback to the user in case of incorrect commands.

Requirements:

- ✓ Use **PJ7 (SEGDP)** for blinking
- ✓ Timer 4 generates periodic interrupts at **100ms**
- ✓ UART configured for non-blocking **DMA** receiving
- ✓ Use **#Command ID*** as command message format
- ✓ Start Command ID = **1**
- ✓ Stop Command ID = **2**

Task C: Read Magnetic Data from LIS3MDL Sensor in Non-Blocking Mode using Interrupts

This task is an extension to task 1 of practicum 3. Previously, a polling method was implemented to read the X, Y, and Z-axis magnetic data from the LIS3MDL sensor. In this task, you are required to **implement interrupts** for I2C read and write. The respective addresses can be found in the datasheet and application note. After reading the magnetic data from the three axes, they must be displayed on a serial monitor of your choice. To do this, you are required to use UART4 of the STM32 MCU with the configuration specified in the tutorial named "**UART Interrupt and DMA Mode**". The read values must be displayed in the following format:

X: *xxxx* Gauss, Y: *yyyy* Gauss, Z: *zzzz* Gauss

Requirements:

- ✓ Use the **I2C2 interface** to read data from the digital sensor
- ✓ Use the **UART4 interface** to send data to the serial monitor
- ✓ Use the **interrupt mode** for reading/writing data
- ✓ Follow the display format mentioned above

Task D: Read Magnetic Data from LIS3MDL Sensor in Non-Blocking Mode using DMA

This task is a modification of the previous task. In this task, you need to use the **DMA mode** of the I2C2 interface for implementing read and write operation.

Requirements:

- ✓ Use the **I2C2 interface** to read data from the digital sensor
- ✓ Use the **UART4 interface** to send data to the serial monitor
- ✓ Use the **DMA mode** for reading/writing data
- ✓ Use the display format mentioned in the previous task