**Cours de 3ème année,**
**Section d'Electricité**

***Last name:*** ……

***First name:***……

***Section:*** ……

***Date and place:***

*Nov. 25th, 2022-MED 22524,MED 22419*

***Duration:*** *1h45*

**Systèmes Embarqués Microprogrammés**

***Grade:*** ……

# *Mid-term Exam*

**IMPORTANT NOTES:**

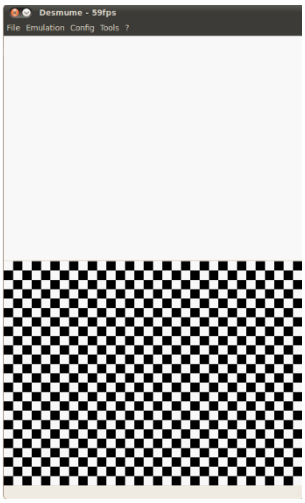
- A skeleton of the exam can be downloaded from the Moodle Site under the link *midterm_2022_2023*. This project is similar to the ones provided during the practical sessions. There are placeholders to implement each exercise of the exam.
- The skeleton of the exam is saved with the project name

"***Midterm_2022_2023_source_code***".

- The exercises must be implemented in the skeleton.
- A compressed file of the project, including the implemented code, must be submitted by means of the form available on the Moodle Site before the end of the exam.
- The implemented exercises must work correctly in the simulator to be considered.
- Follow carefully all the instructions given in the current document and within the comments of the project.

| **EXERCISE 1** | Time: | Works: YES / NO | Sign.: |
|---|---|---|---|

- Configure the bottom screen in tiled mode, activate one single background, and set up a proper VRAM bank.
- Configure the background to display tiled graphics using a standard 32x32 grid with 8-bit pixels.
- 5 custom plain tiles have been declared outside the *main()* function. Copy the tiles to the corresponding place in memory and use the defined *WhiteTile* and *BlackTile* to create the graphic shown in the figure on the right (each little square is a tile, 32x32 pixels in size).
- Initialize the 5 components of the palette with the colors depicted in the figure on the next page (White, Black, Red, Green, and Blue).
- Fill the corresponding map with 24x32 tiles (you must obtain the exact graphic presented in the figure, i.e., top left square is white, and then they alternate between black and white).




| **EXERCISE 2** | Time: | Works: YES / NO | Sign.: |
|---|---|---|---|

- Use the provided image ***logo.png*** and display it on the upper screen of the Nintendo DS. Follow the next steps to fill the corresponding piece of code in the *main()* function in the file *main.c*.
- Configure the MAIN graphical engine in extended rotoscale mode, activate background 2, and set up a proper VRAM bank.
- Configure the background to display graphics at standard 256x256 resolution using a pixel size of 8 bits (therefore using the palette).
- Copy the bitmap and the palette generated by grit to the corresponding places in memory. The grit configuration file is already implemented and included in the project.
  *Note. For Mac OS users, the EPFL logo may appear blue, due to a bug in the grit tool.*


| **EXERCISE 3** | Time: | Works: YES / NO | Sign.: |
|---|---|---|---|

A timer is used to color the WHITE squares in the bottom screen progressively. At a frequency of 20Hz, WHITE squares, from left to right and from top to bottom, should be colored. During the first iteration, all WHITE squares have to become RED iteratively. Once all of them are RED, the same re-coloring approach should be employed to transform all RED squares into GREEN squares. Then, from GREEN to BLUE and finally, from BLUE back again to RED. This color-changing process must be repeated indeterminably. To summarize, all colored squares should iteratively become

RED >> GREEN >> BLUE >> RED >> GREEN >> BLUE >> RED …

BLACK squares remain BLACK. Follow the next steps to complete the corresponding code for this exercise in the file *main.c*:

- In the *main()* function, configure a timer to fire an interrupt 20 times per second (in other words, at 20 Hz).
- Associate the implemented ISR to the interrupt line of the chosen timer after the timer initialization in the *main()* function.
- Enable the corresponding interrupt line of the chosen timer.
  *Note: do not call irqInit() since the touchscreen will be used in the next exercise.*
- Implement the Interrupt Service Routine (ISR) that will be associated to that timer. This function is already declared outside the *main()* function as *timer_ISR()*.

| **EXERCISE 4** | Time: | Works: YES / NO | Sign.: |
|---|---|---|---|

The color update frequency is currently set to 10 Hz. In this exercise, you have to increase or decrease the update frequency when the user presses a certain key. Follow the next steps to complete the corresponding piece of code inside the infinite *while* loop at the end of the *main()* function in the file *main.c.*

- Scan the keys at the beginning of each iteration of the *while* loop and retrieve in a variable the keys that are pressed down.
- If *KEY_UP* is pressed, double the frequency by properly updating the *DATA* register of the employed timer (i.e., *TIMER0_DATA*, if you are using timer 0).
- If *KEY_DOWN* is pressed, halve the frequency by properly updating the *DATA* register of the employed timer (i.e., *TIMER0_DATA*, if you are using timer 0).

| **EXERCISE 5** | Time: | Works: YES / NO | Sign.: |
|---|---|---|---|

Additionally, the color currently filling the squares in the bottom screen can be immediately changed if the user touches the bottom screen (i.e., you don't have to wait for all the squares to update to the same color before changing it).

- Inside the last infinite loop of the *main()* function, also check if the touchscreen has been touched. In that case, continue coloring the remaining squares, but immediately change the current color with the next one. Follow the same order as used for Exercise 3.

| **FINAL TIME**: | Sign.: |
|---|---|