EPFL

Last name:

……

First name:

……

Section:

……

Cours de 3ème année,

Section d'Electricité

Systèmes Embarqués Microprogrammés

Date and place: *November 28th, 2025; MED 2 2524*

Duration: *1h45minutes (from 14h15 to 16h00)*

Grade:

……

Mid-term Exam

IMPORTANT NOTES:

- The skeleton project for the exam can be downloaded from the Moodle Site under the link ***"midterm_code"***. This project is similar to the ones provided during the practical sessions. In the source files, there are **placeholders to implement each exam exercise under the labels "//…TO COMPLETE EXERCISE X"**, indicating where to write code for exercise X, X=1…5. If you find yourself adding code outside the indicated regions, you are definitely doing something wrong or unnecessarily complicated.
- The exercises must be implemented in the skeleton project and completed **in the indicated order**.
- Follow carefully all the instructions given in the current document and the comments of the source code.
- The implemented exercises must work correctly in the NDS simulator to be considered as correctly done. However, the source code will also be evaluated after the exam and must be uploaded to Moodle.
- After finishing **each** exercise, you must call a proctor to check your implementation by marking the time, noting down if the implementation works or not, and signing. From this point onward, **no modifications are allowed.** A compressed file of the project, including the implemented code, must be submitted using the corresponding form available on Moodle for **each** exercise.
- If you fail to complete an exercise, you have the right to ask for its solution to be able to continue with the following exercises.
- The presence in the exam, without completing any exercise, counts as 1 point.

PROJECT DEFINITION:

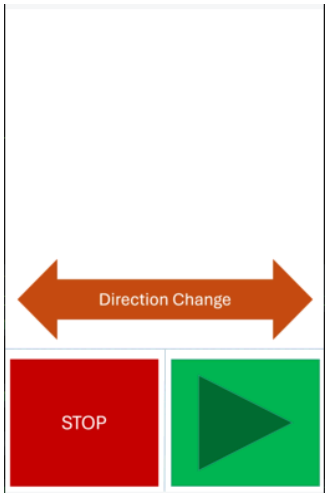
The project consists of 5 exercises to implement a mini-game in which we move a character horizontally along the middle lane of the upper screen. The character is moving based on a timer and can pass through the screen borders and appear on the other edge of the screen. Initially, the character is constantly moving towards the right direction. However, using the keys drawn on the bottom screen through the touchscreen, the character can change movement direction while also changing its orientation. Similarly, the character may stop or resume moving using the stop/play buttons. Try to solve the exercises in the given order by following the detailed instructions and comments in the given code.

EXERCISE 1 (0.75 points)	Time:	C: Y / N, G: Y / N	Teach. Sign.:

The bottom screen will show the game's control keys, namely the direction change for the character movement, and the stop/play buttons. This will be done using an image (keys.png) already provided under the folder *data*. Follow the next steps:

- Uncomment the corresponding code inside *main.c*.
- Complete the function **configureGraphics_Sub()** in the file *graphics.c* following the comments to configure the SUB engine in mode 5 and activate the background BG2.
- Create the configuration grit file inside the *data* folder to obtain the bitmap and the corresponding palette (therefore using 8-bit pixels).
- Complete the function **configBG2_Sub()** in the file *graphics.c* following the given comments to configure the background correctly and transfer the image information to the corresponding locations in memory.
- Compile the project and correct the possible errors.

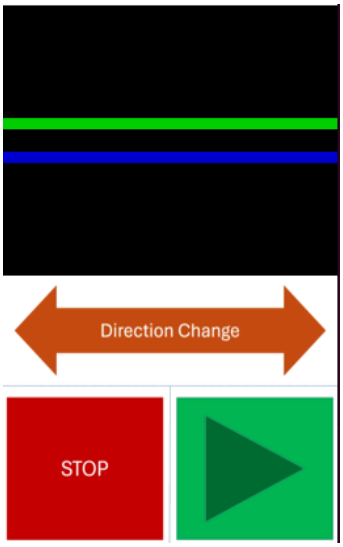
NOTE: the macros related to the SUB engine are followed by the suffix "_SUB" (i.e., BG_TILE_RAM_SUB, BG_PALETTE_SUB, BGCTRL_SUB, etc.).



EXERCISE 2 (1 point)	Time:	C: Y / N, D: Y / N	Teach. Sign.:

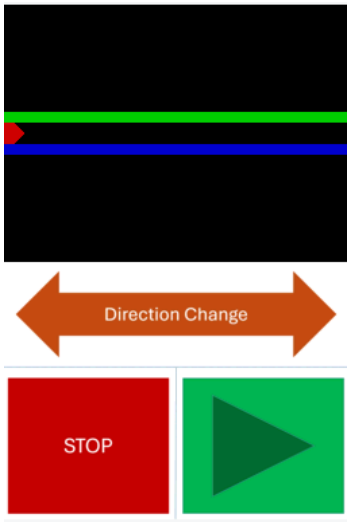
The upper screen will be used to create a lane in the middle of the screen. To do so, the main engine must be configured to work in 16-bit rotoscale mode to draw two horizontal lines. Follow the next steps:

- Uncomment the corresponding code inside *main.c*.
- Complete the function **configureGraphics_Main()** in the file *graphics.c* following the comments to configure the MAIN engine in mode 5 and activate the background BG2.
- Complete the function **configBG2_Main()** in the file *graphics.c* following the given comments to configure the background correctly. NOTE: Set the bitmap starting address 16KB after the start of VRAM_A to save some space for the tiles that will be used later in Exercise 3.
- Draw two horizontal lines across the full screen. The lines have an **8-pixel width** and are placed in a way that they **leave a 16-pixel lane free in the middle of the screen**. The top line is green and the bottom one blue, as indicated in the picture on the right. You can use the color definitions from *colors.h*. The rest of the screen should remain unassigned.
- Compile the project and correct the possible errors.

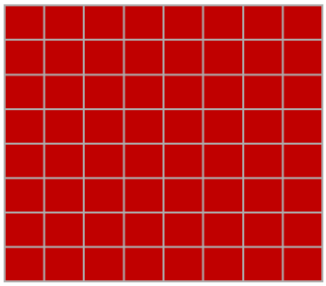


EXERCISE 3 (1.25 points)	Time:	C: Y / N, T: Y / N, D: Y / N	Teach. Sign.:

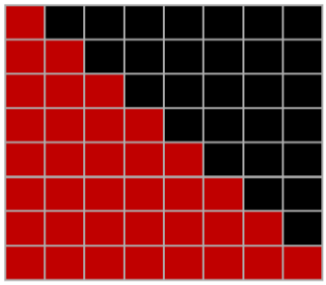
The upper screen will be used to draw the main character. The main engine must be configured to work in tiled mode. The main character (2x2 tiles) will be placed between the two horizontal lines drawn in the previous exercise at the left-most position.
Follow the next steps:



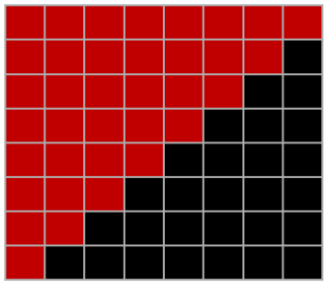
- Uncomment the corresponding code inside *main.c*.
- Ensure that the function **configureGraphics_Main()** in the file *graphics.c* activates both background BG0 and BG2.
- Define the tiles of the main character in the code provided above the function **configBG0_Main()**. As shown below, the character consists of 4 tiles, 2 of which are identical (rear tiles). The front part of the character, consisting of two distinct tiles, forms an arrow that indicates the direction of movement for later exercises. Use the red color from *colors.h*. We will also need an empty tile with transparent pixels for the rest of the screen.
- Complete the function **configBG0_Main()** in the file *graphics.c* following the given comments to configure the background correctly. Then, assign the red color to the palette, copy the tiles to the correct VRAM location and assign the tilemap correctly so that the character appears in the middle lane at the left-most position, as shown in the picture at the right. NOTE: Set the tiles at the start of VRAM_A and the tile-map at 2 KB offset.
- Compile the project and correct any possible errors.



Rear Tile



Front-Up Tile



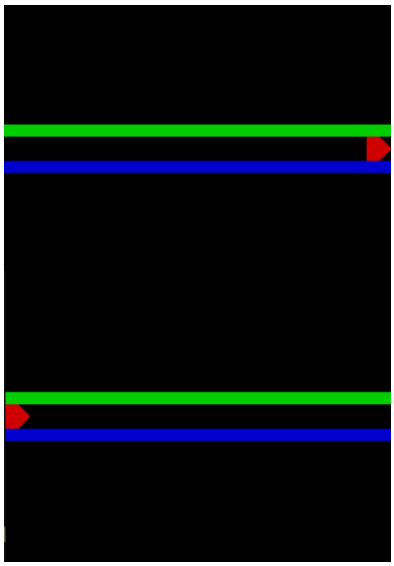
Front-Down Tile

Rear Tile	Front-Up Tile
Rear Tile	Front-Down Tile

Character's Tile layout

EXERCISE 4 (1 point)	Time:	T: Y / N, M: Y / N, E: Y / N	Teach. Sign.:

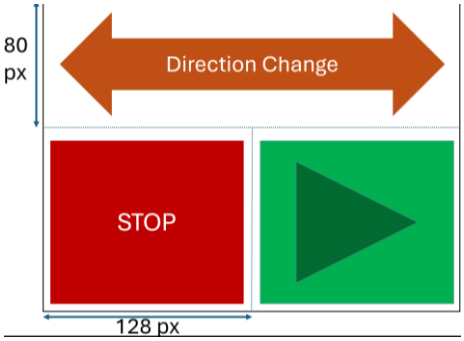
The character can move **horizontally right** across the middle lane. To do so, you must implement a timer and associate it with an ISR that calls the function **movePlayer()**.
Follow the next steps:



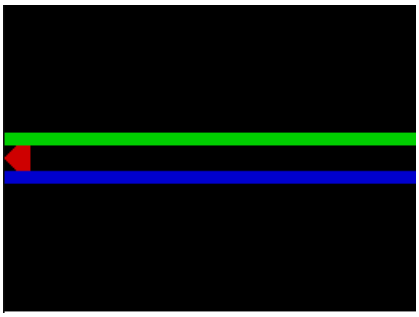
- Uncomment the corresponding code inside *main.c*.
- Complete the code inside *timer.c* to implement a timer that goes off every 125 ms, and associate it with the **timerISR()** function that should call the **movePlayer()** function. Pick a timer that can cover the 125 ms target. NOTE: Do not call **irqInit()** since the touchscreen will be used later.
- Complete the code inside the function **movePlayer()** in *game.c* to update the character's position while moving horizontally to the right with one tile displacement at a time. At the screen edge, the character should pass through and appear on the other screen edge without splitting the character into two parts. The character should transition through the edge as shown in the right pictures.
  HINT: Use the given variables playerX1, playerX2, playerY1, and playerY2, where (X1, Y1) and (X2, Y2) are the tile coordinates of the top-left and bottom-right player's tiles, respectively.
- Compile the project and correct the possible errors.

EXERCISE 5 (1 point)	Time:	T: Y / N, SR: Y / N, DC: Y / N	Teach. Sign.:

In this exercise, the displayed keys of the sub-screen will be utilized to add some functionality. Using the touchscreen, the player can change the movement direction of the character and stop/resume its movement. Follow the next steps:



- Uncomment the corresponding code inside *main.c*.
- Complete the code inside the while loop of *main.c* to handle each of the three displayed buttons and their desired behavior as explained later. The sub-screen is divided into three sections, as shown in the **top right picture**.
- STOP/PLAY buttons: Disable/Enable the timer associated with the character movement. The character must not move after clicking on the STOP button region, and it must always be moving after clicking on the PLAY button.
- CHANGE DIRECTION button: The character must change movement direction (i.e., from right to left, and from left to right). In *game.c*, use the variable **movingDirection** and update the corresponding section inside the **movePlayer()** function. The character should travel to the left with the exact same behavior as moving to the right in Exercise 4.
- For maximum points, ensure that the character tiles are rearranged and the front orientation is pointing to the left, as shown in the **bottom right picture**.
  HINT: Use the horizontal mirror bits provided in Tiled mode.
- Compile the project and correct any possible errors.



PLEASE MAKE SURE YOU HAVE SIGNED AT THE END OF THE EXAM AND ALL THE TEACHER SIGNATURES IN ALL THE EXERCISES SLOTS ARE COMPLETED BEFORE YOU LEAVE THE EXAM, OTHERWISE SOME OF THE EXERCISES MAY NOT BE COUNTED FOR THE FINAL MARK	
FINAL TIME:	Student Sign.: