ULB

BRUFACE
BRUSSELS FACULTY
OF ENGINEERING

VUB

Sensors and Microsystem Electronics

Microcontroller Project

MA1 - Electronics and Information Technology

Oscar Van Slijpe

Academic Year 2021-2022

Contents

1 Overview

The screen is oriented vertically (joystick down). The level containing obstacles is loaded at the top, layer per layer and go down while the controllable player stay at the bottom and can move around to avoid the incoming obstacles.
The screen is taken vertically so this is the orientation used when describing the screen in this report.

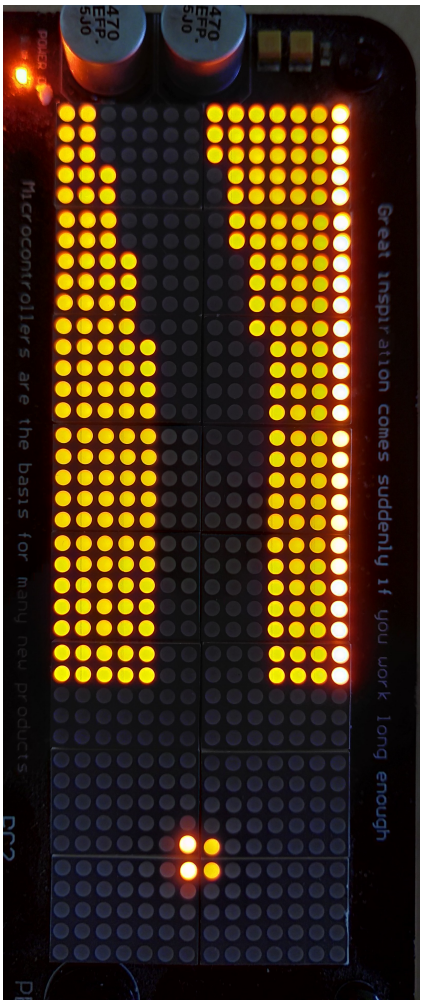


Figure 1: Overview

2 Memory entities

2.1 Layer table

The layer table is the part of the EPPROM memory containing all the layers of the level. Each layer is represented by 2 bytes. Each bit correspond to a column of the screen. The last bit of each bytes is unused since there is only 14 columns. A set bit represent a obstacle and a cleared bit represent a empty space

2.2 Screen Buffer

The screenbuffer is the part of the Flash memory containing the current screen state. Each pixel of the screeen is represented by a bit. The dimension is 16 x 5 bytes to correspond to the screen (with the last 2 column are unusable but still there to match the screen register). The organisation of this buffer match the physical layer of the screen. This is done to make easier the implementation of the game itself but make the code to actually display it a bit more complex.

2.3 Pre Buffer

The prebuffer is and extension of screen buffer conceptually and in term of memory addresses. The dimension is 14 x 1 bytes. It is not actually displayed but it contain the current loaded layer to be shifted to the top of the screen buffer during the progress of the game.

2.4 Player buffer

This is the part of the Flash memory containing the player. This is an array of 14 x 8 bits. The player itself is a square of 2x2 bits set and the remaining values are cleared. This buffer is superposed to the bottom of the screen buffer during the update of the screen where it also check if there is a collision. Moving the player actually shift this buffer (vertically or horizontally).

3 Codes description

3.1 Main

The main loop handle all the keyboard inputs. In order to only trigger once when a key pressed, and to provide a feedback to the user, the led 3 is used to check/display the state of the input. The action associated with a button is only process if the state of the led 3 is off. Once an action is processed, the led 3 is turn on, and it is turn off when the button is not pressed anymore. The input can be divided in two categories: settings and control.

Settings

Those button are in the top two rows and are always available.

- **A/1:** Increase/decrease game speed (Timer 1 initial value)
- **7:** Reset the game
- **0/2:** Increase/decrease layer width.

Control

Those button are in the bottom two rows and are only available when the game is running.

- **D/F:** Moving left/right. The player can loop from left to right, and vice versa.
- **6/E:** Moving top/bot, contains between the 1st and the 8th row from the bottom.

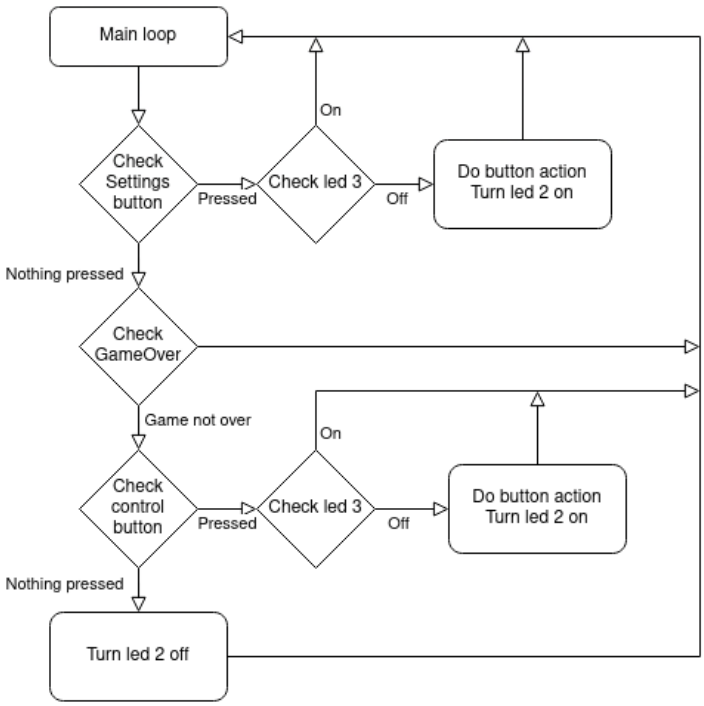


Figure 2: Main loop

3.2 Game update

This is the interrupt of the timer 1. This update the level position. It can be divided in two parts.

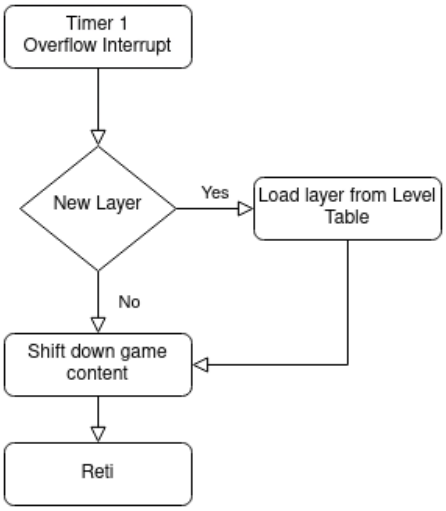


Figure 3: Game update

Load layer

This load the layer from Level table to the pre buffer. It only occurs when the current layer desired width is reached.

Shift game

It shift down each column of the screenbuffer and copy the content of the prebuffer into the top row of the screen buffer.

3.3 Display update

This is the interrupt of the timer 0. It read the value from screen buffer and player buffer and display it on the screen.
The main issue is the fact that those buffer are organised to match the physical layout of the screen, which is not the same as the electronic layout. The figure 4 shows the point of view with the numbering corresponding the to corresponding index of the screenbuffer, the idea is the same for the player buffer.

1	-->	5
6	-->	10
11	-->	15
16	-->	20
21	-->	25
26	-->	30
31	-->	35
36	-->	40
41	-->	45
46	-->	50
51	-->	55
56	-->	60
61	-->	65
66	-->	70
71	-->	75
76	-->	80

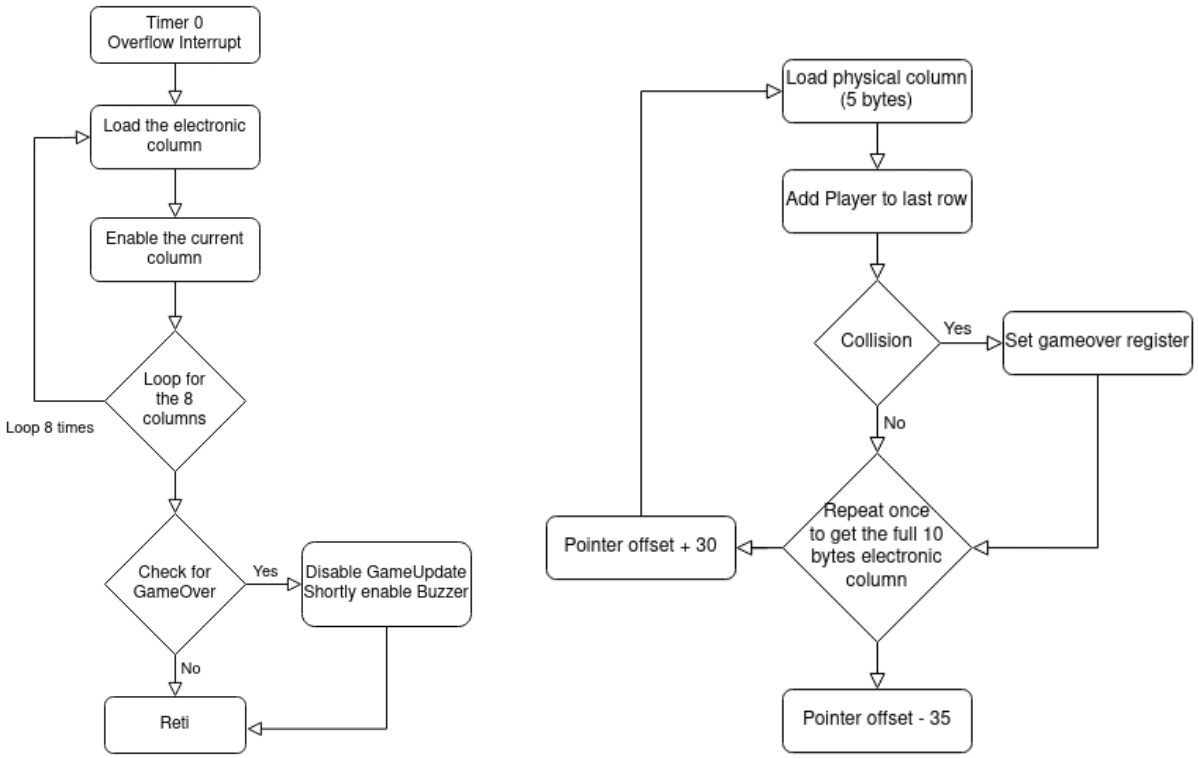
(a) Physical

1	-->	5	36	-->	40
6	-->	10	41	-->	45
11	-->	15	46	-->	50
16	-->	20	51	-->	55
21	-->	25	56	-->	60
26	-->	30	61	-->	65
31	-->	35	66	-->	70
71	-->	75	76	-->	80

(b) Electronic

Figure 4: Screen layout

This mean than instead of loading the full 10 bytes (electronic column), it had to load 5 bytes (right physical column), then add an offset (30) to the pointer, load the 5 next bytes (left physical column) and finally remove the offset from the pointer (35).



(a) Global (b) Column loading

Figure 5: Display flow

When loading the last bytes of each physical column, it also load the corresponding column of the player buffer and add it (with "OR") to the current column, to display the player on top of the game. It also check if there is a collision between the game and the player. If this is the case, it set led 2 on, disable the game interrupt, and shortly enable the buzzer interrupt. The led 2 is turn off when resetting the game.

3.4 Buzzer

This is the interrupt of the timer 2. When enable, is simply toggle the buzzer to get the annoying sound, hopefully only for a short period of time.

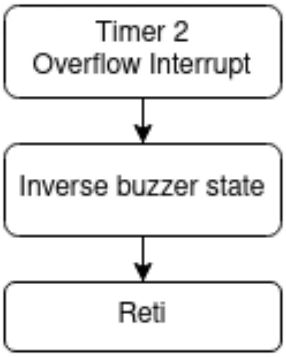


Figure 6: Buzzer