Semester work APO 2024

“Game of life”

Programmer's manual

Author: Oleksii Kolesnichenko

e-mail: kolesole@fel.cvut.cz

0. GitLab

For version control was used GitLab:

Link

1. knobs.h/c

knobs.h - header file with functions to read a knobs state;

knobs.c - source file with functions to read a knobs state;

- used libraries:
  - <stdint.h>;
  - “mzapo_regs.h”;
- functions:
  - uint32_t filtered_knobs_state(unsigned char* spi_mem_base):
    - function that returns filtered knobs state;
  - uint32_t red_knob_state(unsigned char* spi_mem_base):
    - function that returns red switch state;
  - uint32_t green_knob_state(unsigned char* spi_mem_base):
    - function that returns green switch state;
  - uint32_t blue_knob_state(unsigned char* spi_mem_base):
    - function that returns blue switch state;
  - uint32_t red_button_state(unsigned char* spi_mem_base):
    - function that returns red button state;
  - uint32_t green_button_state(unsigned char* spi_mem_base):
    - function that returns green button state;
  - uint32_t blue_button_state(unsigned char* spi_mem_base):
    - function that returns blue button state;
  - void light_up_led_line(unsigned char* spi_mem_base):
    - function that lights up led line;
  - void go_out_led_line(unsigned char* spi_mem_base):
    - function that turns off led line;
  - void light_up_red_color_rgb1(unsigned char* spi_mem_base):
    - function that lights up rgb1 in red;
  - void go_out_rgb1(unsigned char* spi_mem_base):
    - function that turns off rgb1;
  - void light_up_green_color_rgb2(unsigned char* spi_mem_base):
    - function that lights up rgb2 in green;

- void go_out_rgb2(unsigned char* spi_mem_base):
  - function that turns off rgb2.

2. cell.h/c

cell.h - header file with functions that perform all related actions with cells;

cell.c - source file with functions that perform all related actions with cells;

- used libraries:
  - <stdbool.h>;
  - <stdlib.h>;
  - “knobs.h”;
- structure:
  - name: cell;
  - elements:
    - bool live (if the cell is alive live = true);
    - int quantity_of_neighbors (the number of living cells around a given cell);
- functions:
  - void set_cell(cell* cells_array[32][48], int raw, int column):
    - function that runs around a cell and for the cell on the position (raw, column) checks if the cells on the positions (raw - 1, column - 1), (raw - 1, column), (raw - 1, column + 1), (raw, column - 1), (raw, column + 1), (raw + 1, column - 1), (raw + 1, column), (raw + 1, column + 1) are alive;
  - bool cell_birth(unsigned char* spi_mem_base, int cursor_column, int cursor_row, cell* cells_array[32][48]):
    - function that revives the cell and returns true if red switch is pressed;
  - void cell_die(unsigned char* spi_mem_base, int cursor column, int cursor_row, cell* cells_array[32][48]):
    - function that kills the cell on the position (cursor_row, cursor_column) if blue switch is pressed and this cell is alive;
  - bool set_start_cells_array(cell* cells_array[32][48):
    - function that allocates memory for the entire cells_array, configures each cell (live = false, quantity_of_neighbors = 0) and returns true if it succeeded;
  - void set_cells_array(cell* cells_array[32][48):
    - function that revives or kills cells depending on the rules;
    - rules:

1) if the cell is alive & quantity of neighbors < 2 or > 3 ----> the cell dies;
2) if the cell is dead & quantity of neighbors = 3 ----> the cell is reborn;

- void delete_cells_array(cell* cells_array[32][48]):
  - function that deletes allocated memory for cells_array;
- bool get_population_growth(cell* cells_array[32][48], int* num_of_cells):
  - function that calculates new_num_of_cells & return true if new_num_of_cells >= num_of_cells;

3. cursor.h/c

cursor.h - header file with function to determine the location of the cursor;

cursor.c - source file with function to determine the location of the cursor;

- used libraries:
  - <stdbool.h>;
  - “knobs.h”;
- functions:
  - void set_cursor(unsigned char* spi_mem_base, int* cursor_column, int* cursor_row, int* red_knob_last_state, int* blue_knob_last_state):
    - function that calls a functions cursor_horizontal_movement(-----//-----) & cursor_vertical_movement(-----//-----);
  - void cursor_horizontal_movement(unsigned char* spi_mem_base, int* cursor_column, int* red_knob_last_state):
    - function that adjusts the horizontal cursor position depending on the position of the red switch;
    - if red_knob_state is at the limit value (255/0) return;
    - if red_knob_new_state + 1 < red_knob_last_state cursor moves to the left;
    - if red_knob_new_state > red_knob_last_state + 1 cursor moves to the right;
    - 1 is added for lower sensitivity;
  - void cursor_vertical_movement(unsigned char* spi_mem_base, int* cursor_row, int* blue_knob_last_state):
    - function that adjusts the vertical cursor position depending on the position of the blue switch;
    - if blue_knob_state is at the limit value (255/0) return;

- if blue_knob_new_state + 1 < blue_knob_last_state cursor moves to the up;
- if blue_knob_new_state > blue_knob_last_state + 1 cursor moves to the down;
- 1 is added for lower sensitivity;

4. menu.h/c

menu.h - header file with function for set menu;

menu.c - source file with function for set menu;

- used libraries:
  - “cell.h”
- functions:
  - void set_menu(cell* cells_array[32][48]):
    - function that configures menu_array;
    - revives the cells from which the inscription PLAY and READY LAYOUT are obtained;

5. ready_layouts.h/c

ready_layouts.h - header file with function to set cells_array after selecting a ready layout;

ready_layouts.c - source file with function to set cells_array after selecting a ready layout;

- used libraries:
  - “cell.h”;
- functions:
  - void set_ready_layouts(cell* cells_array[32][48]):
    - function that calls all functions from this file for creating ready_layout_array;
  - void set_glider(cell* cells_array[32][48]):
    - function that configures cells_array to the glider;
  - void set_pulsar(cell* cells_array[32][48]):
    - function that configures cells_array to the pulsar;
  - void set_spaceship(cell* cells_array[32][48]):
    - function that configures cells_array to the spaceship;
  - void set_glider_gun(cell* cells_array[32][48]):

- function that configures cells array to the glider gun;

6. scene.h/c

scene.h - header file with functions that set stage;

scene.c - source file with functions that set stage;

- used libraries:
  - <stdbool.h>;
  - “mzapo_parlcd.h”;
  - “menu.h”;
  - “ready_layouts.h”;
  - “cell.h”;
  - “cursor.h”;
  - “knobs.h”;
- defined:
  - DISPLAY_WIDTH 480;
  - DISPLAY_HEIGHT 320;
  - WHITE 0xFFFF (dead cell membrane color) ;
  - BLACK 0x0000 (dead cell color);
  - RED 0xF800 (color of a cell that is alive);
  - YELLOW 0xFFE0 (cursor color);
- functions:
  - void set_loading_scene(unsigned char* parlcd_mem_base,
    cell* cells_array[32][48]):
    - function that turns on the loading scree;
    - background of cells appears ant the cells from the four corners are gradually revived;
  - void set menu_sceen(unsigned char* parlcd_mem_base,
    cell* cells_array[32][48],
    int* green_knob_last_state,
    unsigned char* spi_mem_base,
    bool* button play):
    - function that turns on the menu;
    - depending on the change in the green switch, the function draws a white rectangle on the play or ready layout inscription;

- void set_ready_layouts_scene(unsigned char* parlcd_mem_base,
  cell* cells_array[32][48],
  int* green_knob_last_state,
  unsigned char* spi_mem_base,
  int* cursor_state):
  - function that turns on the ready layouts scene;
  - configures the array and depending on the change in the green switch, the function draws a white rectangle around ready layouts;
- void set_scene(unsigned char* parlcd_mem_base, cell* cells_array[32][48],
  int cursor_column, int cursor_row):
  - function that sets up the scene during the game;
  - draws a cursor and new living or dead cells;

7. game_of_life.h/c

game_of_life.h - main header file;

game_of_life.h - main source file;

- used libraries:
  - <stdio.h>;
  - <unistd.h>;
  - <time.h>;
  - <bits/types/struct_timespec.h>;
  - “mzapo_phys.h”;
  - “scene.h”;
  - “cell.h”;
  - “knobs.h”;
  - “cursor.h”;
- functions:
  - void my_sleep():
    - function that puts the game into sleep mode for 0.35 seconds;
  - bool allocate_memory(cell* cells_array[32][48],
    cell* menu_cells_array[32][48],
    cell* ready_layouts_cells_array[32][48]):
    - function that allocates all necessary memory for the game and returns true if everything was successful;
  - void delete_allocated_memory(cell* cells_array[32][48],

cell* menu_cells_array[32][48],
cell* ready_layouts_cells_array[32][48]):

- function that delete all alocated memory;

- bool map_phys_addresses(unsigned char* parlcd_mem_base,
unsigned char* spi_mem_base):
  - function that maps to all necessary addresses for the game(parlcd & spi);
- void start_menu_mode(unsigned char* parlcd_mem_base,
cell* menu_cells_array[32][48],
unsigned char* spi_mem_base,
bool* button_play):
  - function that displays a menu and updates it;
  - menu is constantly updated until the green button is pressed;
- void start_selection_mode(unsigned char* parlcd_mem_base,
cell* cells_array[32][48],
unsigned char* spi_mem_base,
int* cursor_column,
int* cursor_row):
  - function that displays selection mode, lights up led line and updates it until the green switch is pressed;
- void start_play_mode(unsigned char* parlcd_mem_base,
cell* cells_array[32][48],
unsigned char* spi_mem_base,
int* cursor_column,
int* cursor_row):
  - function that displays play mode and updates it;
  - if the green switch is pressed it returns to the menu;
  - if the red switch is pressed it return to the selective mode;
- void start_play_mode(unsigned char* parlcd_mem_base,
cell* ready_layouts_array[32][48],
unsigned char* spi_mem_base,
int* cursor_state):
  - loading ready layout scene and updates it until the green button is pressed;
- int main(int argc, char* argv):
  - the main function that ties the game together using an infinite loop.