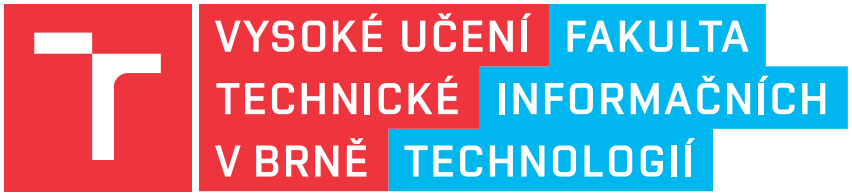


# Project dokumentation

## ARMFITkit3 či jiný HW: Hra HAD

IMP Microprocessors and Embedded Systems

December 27, 2022

Zdeněk Lapeš (xlapes02)
*lapes.zdenek@gmail.com*

# Contents

# 1 Introduction

The task was to create a Snake Game for the ARMFITkit3 board based on microcontroller Kinetis K60 (with ARM CortexM4 core) and 2x matrix displays (type: KWM30881AGB, decoder: 74HCT154).

The result is a Snake game written in C language using KDS IDE[**?**]. Snake is displayed on the matrix displays and the player can control the snake using the *Fitkit3* 5 builtin buttons.

# 2 Preparation

## 2.1 Hardware

The provided HW for this project is *Fitkit3* board and one board with 2x matrix displays. The board with 2 matrix displays is connected to the *Fitkit3* board using the connectors P1 (placed on the Fitkit3 board) and connector P3 (placed on the matrix display's board). One matrix display is (8,8) so the total size of the display is (16,8).

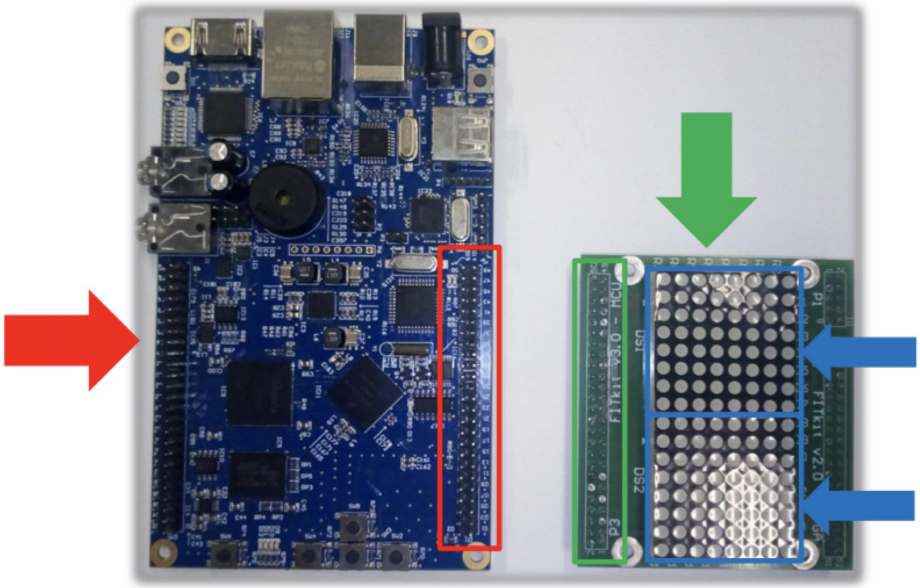

Figure 1: Fitkit3 board & matrix display board

# 3 Implementation

Implementation of the game is inside file: `Sources/main.c`. All other files were created by the KDS IDE for MCU K60.

The entry point of the program is the function `main()` located in `Source/main.c` file. At the beginning of the program, the function `SystemConfig()` is called where the peripherals of MCU are configured:

- Configuration of PTA's pins (GPIO) for the matrix displays.
- Configuration of PTE's pins (GPIO) for the buttons.
- Configuration of PIT (used for timer).
- Timer and Button interrupts are enabled.

After MCU configiration is `snake_t` structure is initialized. This structure contains all information about the snake: **body**, **direction**, **speed**.

Then the game starts by continue to endless loop, where all incoming interrupts by **PIT** timer and buttons are handled, using the functions `PIT0_IRQHandler()` and `PORTE_IRQHandler()`.

### 3.1 Timer interrupt Handler:

Snake next position is calculated here and display is updated by calling one of the functions: `move_up()`, `move_down()`, `move_left()`, `move_right()`.

```
void PIT0_IRQHandler() {
    int column_number = i % 4;

    // Clear the timer's interrupt flag
    PIT->CHANNEL[0].TFLG = 0x1;

    // Set new snake body position
    if (snake.direction == RIGHT && TIMER_CHANGE_OFSET(i)) {
        move_right();
    } else if (snake.direction == UP && TIMER_CHANGE_OFSET(i)) {
        move_up();
    } else if (snake.direction == DOWN && TIMER_CHANGE_OFSET(i)) {
        move_down();
    } else if (snake.direction == LEFT && TIMER_CHANGE_OFSET(i)) {
        move_left();
    }

    // Place snake onto LED display
    ENABLE_LED_WRITE;
    if (column_number == 0) { PTA->PDOR = snake.head; }
    else if (column_number == 1) { PTA->PDOR = snake.body1; }
    else if (column_number == 2) { PTA->PDOR = snake.body2; }
    else if (column_number == 3) { PTA->PDOR = snake.tail; }
    DISABLE_LED_WRITE;

    //
    i++;
}
```

### 3.2 Button interrupt Handler

Snake direction, speed and position is changed here according to the pressed button.

```
void PORTE_IRQHandler(void) {
    // Wobble
    if (IS_BTN_INTERRUPT_WOBBLE == 0) {
        // Clear interupt flags
        PORTE->ISFR = BTNs_ALL_MASK;
        return;
    }

    delay(tdelay1 / 100, 1); // Filtering wobble

    // BTN_DOWN
    if (IS_CLICK_DOWN && snake.direction == UP) {
```

```
        snake.direction = RIGHT;
    } else if (IS_CLICK_DOWN && snake.direction == DOWN) {
        snake.direction = LEFT;
    } else if (IS_CLICK_DOWN && snake.direction == LEFT) {
        snake.direction = UP;
    } else if (IS_CLICK_DOWN && snake.direction == RIGHT) {
        snake.direction = DOWN;
    }

    // BTN_UP
    if (IS_CLICK_UP && snake.direction == UP) {
        snake.direction = LEFT;
    } else if (IS_CLICK_UP && snake.direction == DOWN) {
        snake.direction = RIGHT;
    } else if (IS_CLICK_UP && snake.direction == LEFT) {
        snake.direction = DOWN;
    } else if (IS_CLICK_UP && snake.direction == RIGHT) {
        snake.direction = UP;
    }

    // Speed
    if (IS_CLICK_RIGHT && snake.speed > 50) {
        snake.speed -= 50;
    } else if (IS_CLICK_LEFT && snake.speed < 350) {
        snake.speed += 50;
    }

    // Restart game
    if (IS_CLICK_START_STOP) {
        init_snake_body_variables();
    }

    // Clear interupt flags
    PORTE->ISFR = BTNs_ALL_MASK;
}
```

# 4 Functionality & Game Control

## 4.1 About the Game

The game is a single player game.

### 4.1.1 Game Control

The game is controlled by the *Fitkit3* builtin buttons:

- **SW2** Snake speed up.
- **SW3** Snake turn right.
- **SW4** Snake speed down.
- **SW5** Snake turn left.
- **SW6** Snake reset (default speed and starting position).

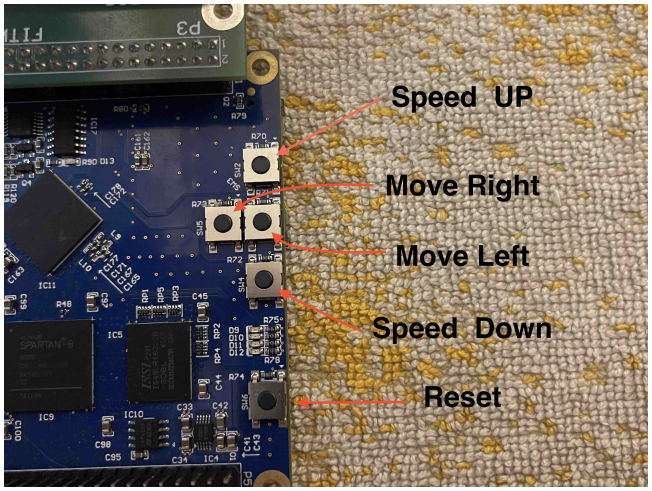


Figure 2: Fitkit3 Buttons

The Snake starts moving to the right with normal speed at the beginning of the game. By pressing the coressponding button, you can change the snake speed and direction. If you want reset the game, press the button **SW6**, which sets the speed to the default value and places snake to the starting position.

## 5 Demonstration of implemented game

The game is demonstrated in the video on this url: **https://youtu.be/V0ONHRM66mM**.

## 6 Conclusion

The project was successfully completed and the game is working as expected. I managed to implement whole functionality of the game with some additional features like speed up/down and reset**??**. In project I used the timer interrupt for controlling the snake moves and button interrupts for controlling the snake direction and speed.

# 7 Autoevaluation

Table 1: Autoevaluation

| Task | Points | Description |
|---|---|---|
| **E** | 1 | I began long before the deadline and afterward I needed to fix some special cases, what I have learned from laboratories. |
| **F** | 5 | The whole functionality requirements were covered. |
| **Q** | 3 | Code should be straightforward to understand almost everybody. |
| **P** | 1 | Illustration of functionality can be watched on youtube click here to watch |
| **D** | 4 | All documentation requirements are covered. |
| **Total** | 14 | |