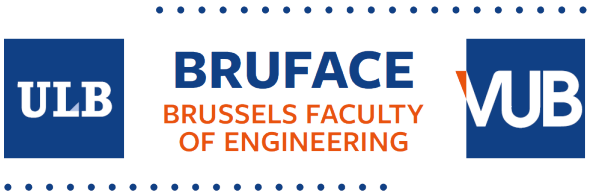

ELEC-Y418: Sensors and Microsystem electronics

Microcontroller Project
Mario Tiles

Jian HUO, Glodi SALA MANGITUKA

Supervised by
Tuur Bruneel
Ayman Morsy

Academic year
2023 - 2024

Faculty
Electronics Engineering

Contents

1 Introduction **2**

2 User Guide **2**

2.1 Menu 2

2.2 Free mode 2

2.3 Level mode 2

3 Implementation **2**

3.1 Screen 2

3.2 Keyboard 3

3.3 Joystick 3

3.4 Switch 3

3.5 Buzzer 3

3.6 Timer2 3

3.7 Tables 3

4 Block Diagram **3**

5 Conclusion **3**

1 Introduction

This project showcases the development of "Mario Tiles," a piano game coded in assembly language. The game offers two engaging modes: "Free Mode," where players can enjoy playing tunes using the keypad as piano keys. Furthermore, "Level Mode," which challenges players to match displayed keys. Success allows progression through the level, while failure results in a score display. This report elucidates the principles of each mode, presenting the project scheme and user interface. Additionally, it delves into the coding structure, elucidating the game's mechanics and highlighting its interactive features.

2 User Guide

The user guide outlines the project's behavior, detailing how users should interact with it. It provides clear instructions on navigating between modes, playing the piano, and understanding the game mechanics.

2.1 Menu

Upon powering on the microcontroller, the switch (PB0) has to be in the high position. The menu will appear on the screen, accessible via joystick (AD0) movements—up and down for navigation. Selection of either the Free mode or Level is accomplished by pressing the joystick (PB2) with the option to exit a mode available by pressing the joystick once more, returning to the main menu.

2.2 Free mode

In the first mode, users can engage by pressing the keys of the keyboard (PD 0-7). Each key press activates the buzzer (PB1), emitting distinct notes corresponding to the pressed key, allowing for interactive musical expression.

2.3 Level mode

In the second mode, the screen guides players by displaying the key to press for gameplay. Two scenarios lead to failure: pressing incorrect keys (PD0-PD7) that don't match the displayed ones or exceeding the time limit of 3 seconds per key press, triggering a game over screen. However, successfully pressing the correct keys prompts continuation to the level, accompanied by a winning sound. Regardless of outcome, the user's score, tallying the number of correctly pressed keys, is prominently displayed before returning to the main menu.

3 Implementation

The project's implementation involves explaining the function of each component and how they're triggered. Additionally, it will describe the constants, tables, and macros used in the assembly code.

3.1 Screen

The screen divides into blocks using predefined characters, each block's pattern read from SRAM memory at specific addresses. Subsequently, the data for each block shifts to the right line for display. Macros like SCREEN_X write data for each block in SRAM to construct various displays (such as the menu, free mode, level, and game over). Additionally, the LOAD_CHARACTER macro enables the retrieval of data corresponding to different blocks by reading the correct SRAM address.

3.2 Keyboard

The keypad consists of 16 different keys. The macro CHECK_KEYBOARD implements a two-step algorithm to determine whether the user has pressed a key or not.

3.3 Joystick

The joystick features two internal interrupts: SW_ISR and ADC_ISR. SW_ISR is triggered when the joystick is pressed, leading to a change in the screen selection register or resetting to the menu. On the other hand, ADC _ISR reads the ADC value to determine the position of the menu arrow. Additionally, when the joystick is moved up or down, the LED PC3 is activated.

3.4 Switch

The switch serves to power off and on the microcontroller. An internal interrupt is triggered when the switch changes its state, and its new state is stored in register r0. The macro CHECK_SWITCH allows for the microcontroller to be switched off when the switch is in a low state.

3.5 Buzzer

The buzzer is utilized in conjunction with the Timer0_ISR interrupt. The timer's loaded value, determining its frequency, depends on the notes to be played. Frequency values for each note are computed in the constants. Additionally, when the t flag is set, the buzzer is enabled to produce sound at the frequency defined by Timer0.

3.6 Timer2

Timer2 serves as a constant timer, measuring intervals in increments of 100 milliseconds. It tracks centiseconds, deciseconds, and seconds, providing accurate timing for various operations such as playing notes during the level, game over, and winning sound sequences.

3.7 Tables

The code includes four tables. The first, TableChar, encodes characters for screen blocks, each consisting of 7 lines and 5 columns. The second table pertains to the Mario game, pairing keys with corresponding frequency notes to play and their associated durations. Lastly, there are tables for the winning and game over sounds, listing the notes and their respective durations.

4 Block Diagram

5 Conclusion

In conclusion, "Mario Tiles" showcases the intricacies of assembly language coding with its two engaging modes: "Free Mode" and "Level Mode." This report provides a comprehensive overview of the project's design, user interface, and implementation details, including keypad recognition, joystick controls, and sound generation. Through clear instructions and detailed explanations, this project offers valuable insights into embedded programming and interactive gaming development.

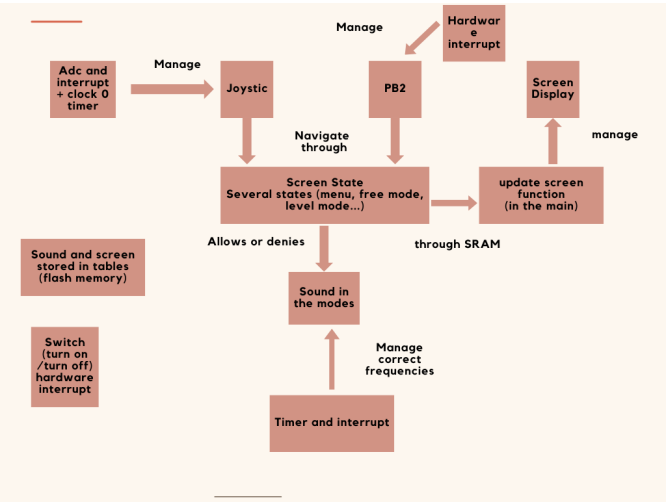

Figure 1: Block Diagram