EE 306-03: Microprocessors

Term Project - Report

« DnDicer »

Bogdán Itsám Dorantes-Nikolaev - 042101002

Onur Keleş - 042001049

Ömer Mert Yıldız - 042002008

Department of Engineering, MEF University

Prof. Tuba Ayhan

May 17th, 2024

1. Abstract

This ARM assembly program operates as a digital dice roller on a microcontroller platform, interfacing with hardware peripherals. It supports 11 distinct dice configurations: 2-sided, 4-sided, 6-sided, 8-sided, 10-sided, 12-sided, 20-sided, 32-sided, 64-sided, and 99-sided. Users select the dice type via switches, with each switch corresponding to one dice type. The program includes error checking, displaying "Er" on the HEX display for multiple toggles and "SL" for no selection. The ready state or initial display is indicated by "rdy" on the HEX display. A timer provides seed values for the random number generator, with results shown on HEX displays, ensuring clear and immediate visual feedback of the dice roll outcome.

2. Flowchart

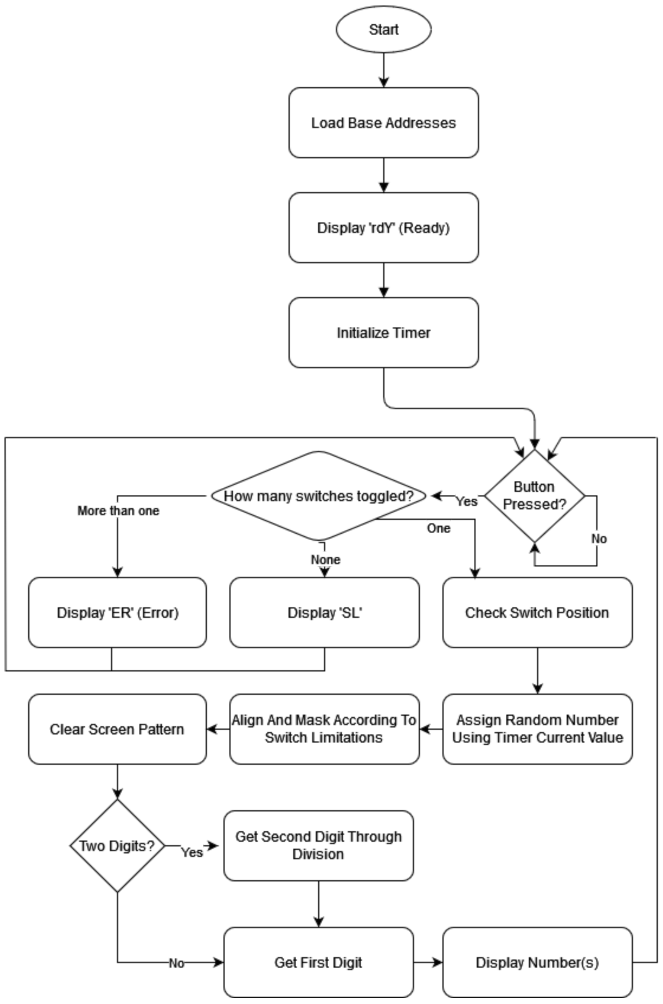

Figure 1. Flowchart of "DnDicer"

3. User Manual

Normal Usage:

Upon launch, the device prompts that it is in a ready state:

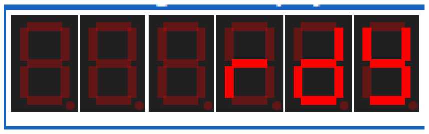

Figure 2. Ready message indicating a successful program launch.

Choose a desired dice (or coin) using the toggle switches:

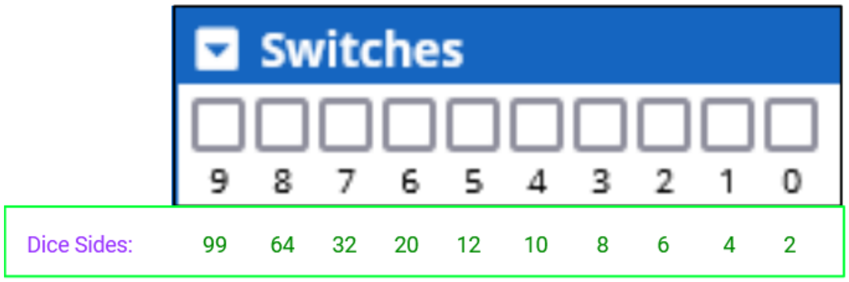

Figure 3. Toggle-switch legend for their corresponding dice side values.

After selecting a dice, use any of the push-buttons to roll the dice:

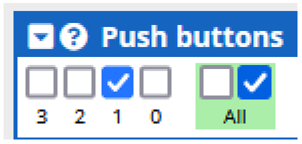

Figure 4. Push-buttons that perform the dice-rolling.

A usage example for the common 6-sided dice:

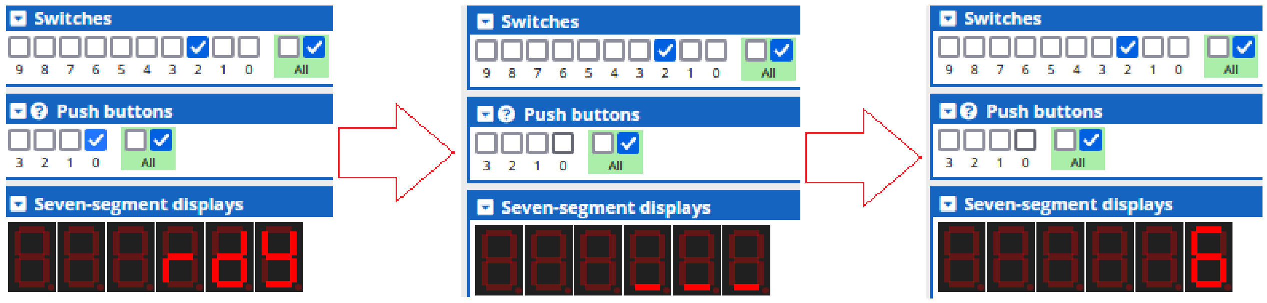

Figure 5. Program set to 6-sided dice and generated the value of 6.

Abnormal Usage:

Duplicate Switching:

When more than one switch is toggled and the dice are rolled using the push button, the device will prompt an error message:

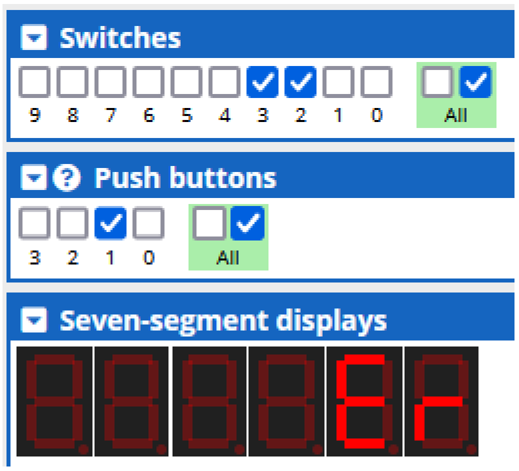

Figure 6. Error indication when more than one switch is toggled and dice rolled.

Undefined Switching:

When no switches are toggled and the dice are rolled using the push button, the device will prompt the user to make a selection:

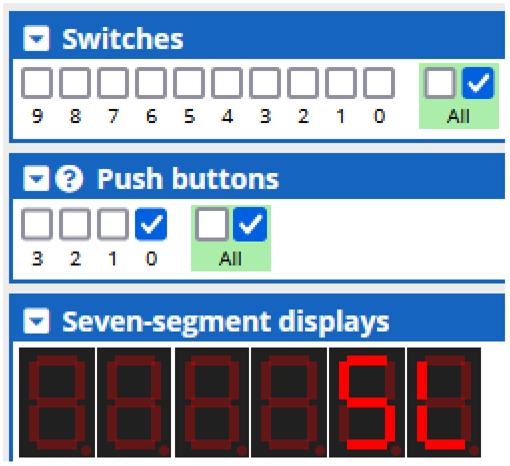

Figure 7. Selection prompting when no switches are toggled and dice rolled.

4. Tackled Problems

Random Number Generation:
Our team began by exploring various techniques for creating random numbers. Initially, we experimented with exclusive-OR (EOR) operations and a seed value derived from a register. But after some trials, we decided to use a private timer instead. This choice was made because the timer provided a more reliable and efficient way to achieve randomness.

Switch Addressing:
While working on assigning each switch to correspond with a specific dice, we ran into a bit of a snag with the incrementation steps. At first, we tried lining up the switch addresses with each potential position through direct comparison. However, this led to some hiccups during the incrementation phase. We found a neat workaround by shifting the comparison value instead of multiplying it, which smoothly clarified the issues. This method ensured flawless switch addressing and enabled precise dice selection without any unexpected complications.

7-Segment Display Addressing:
Our first attempts at showing the randomly generated numbers faced a hurdle: they were limited to the rightmost segment of the HEX display. This became problematic when the numbers extended beyond two digits, mostly due to the differences between decimal and binary representations. To tackle this, we introduced a digit counter mechanism. By deducting the base decimal value from our random number, we could figure out the necessary number of digits and adjust the display accordingly. This clever strategy allowed us to display multi-digit numbers effectively on the HEX display, significantly enhancing the system's capability and user experience.