Dodge Virus

Microcontroller project

Boya Zhang

July 8, 2020

Engineering Science

1 Introduction

In this project, we implemented a Dodge Virus Game on ATmega328P micro-controller using AVR Assembly. It can detect the input of keyboard and show changes on the screen.The buzzer rings and the bottom LED lights up when we press buttons under certain conditions.

2 User Guide

To play with dodge virus, users can follow these steps:

1. **Game cover:** At first you can see a front page given the name dodge virus with pattern of Steve and the virus, press button 5 to start the game process.
2. **Game process:** At this part you take control of Steve to walk to the exit. If during the process you touched the virus, then the sign "YOU LOSE" will show up. On the other hand, if you make it to the exit without touching any virus, then the sign "YOU WIN" will show up.
3. **Restart:** If you want to restart, press button 5 and go back to the initial position to play it again.
4. **Button and boundary:**
   (1) Press button 8,2,4,6 to go up, down, left or right. Press button 5 to go to the start position or to restart the game.
   (3) If the user tries to move Steve out of the screen, it can be detected. Steve doesn't move and the buzzer rings.
   (4) If the user press buttons that are useless, buzzer rings and bottom LED light up.

3 Core part

3.1 Buzzer

We take tone "la" at 440Hz as the ring frequency. It requires an interrupt at 880Hz. We use timer0 which is 8 bits. The overflow value is 256.

$$f_{clk} = \frac{16MHz}{256} = 62500 \tag{1}$$

$$TCNT_{init} = 256 - \frac{62500}{880} = 185 \tag{2}$$

The correct prescalar 256 we choose gives us the exact frequency with no overflow or round error.
As in the display we have a delay in the "Enable" function, the frequency of the buzzer is lower than 440Hz when button is pressed. And when "Hold" is called the buzzer doesn't sound because the delay time is rather long.
Function "Hold" is not called in function "Do Nothing", so the buzzer will ring at a frequency lower than 440Hz when the press doesn't meet boundary constraint.

3.2 Screen Display

We divide the screen into 16 blocks each with 7 rows (8 bits in the memory, last bit is set to zero) and 5 columns. From address 0x100 to 0x10F, we store the charbuffer for each block. Each charbuffer refers to one character defined in the chartable.

3.2.1 Algorithm

The charbuffer loop contains the following steps. First, we get buffer value from the memory. Second, we calculate the address for the character. Third, we get pattern from chartable. Next,we shift out.
The row loop contains the charbuffer loop, row selection, latch and output enable.

3.2.2 Calculation

From address 0x10F to 0x100, we read out different value of charbuffer and store it in Rx. The address for each character is $ADR = Chartable + 8Rx + Row$.

3.3 Game Control

In the game control part we use charbuffer to control the game. When the charbuffer value in the address is controlled, the character is automatically controlled with the screen display part.

3.3.1 Algorithm

We use the 4-steps method to check the button pressing. When a move button is pressed, we read out the position of Steve through two registers. One stored higher 8 bits, the other stored lower 8 bits. Then 1) deduct or add 8 to the pointer when pressing up or down. 2) deduct or add 1 to the pointer when pressing left or right. Then we store new Steve position to the registers. For the previous position, the pattern is set to empty. For the new position, the pattern is set to Steve.

3.3.2 Certain conditions

- Under encountering virus or exit condition, we jump to "LOSE" or "WIN" and then restart.
- Under boundary condition, the result of the pressing should be Steve not moving.
  - We know that ZH doesn't change and is always 0x01.
  - When Steve goes left or up outside the screen, ZL goes under 0x00. Because of the overflow, the value for ZL is bigger than 0x0F.
  - When Steve goes right or down outside the screen, ZL goes above 0x0F.
  - So when the new Steve position is outside the screen, the value for ZL is always bigger than 0x0F. We take it as a constraint to call function "Do Nothing".
  - At the "LEFT" function, when Steve is at the bottom-left corner we also call function "Do Nothing". The same is "RIGHT" function at up-right corner.

4 Conclusion

1. We use the "Hold" function to make the human press equals one press to the microcontroller. Inside the "Hold" function we should also put the screen display so that the screen doesn't blink when we press the button. And be careful with the register to not cause infinite loop.
2. The game needs to be separated into two parts: game control and screen display. The bridge between them are charbuffer and chartable. It is the same idea for nowadays computer game design.

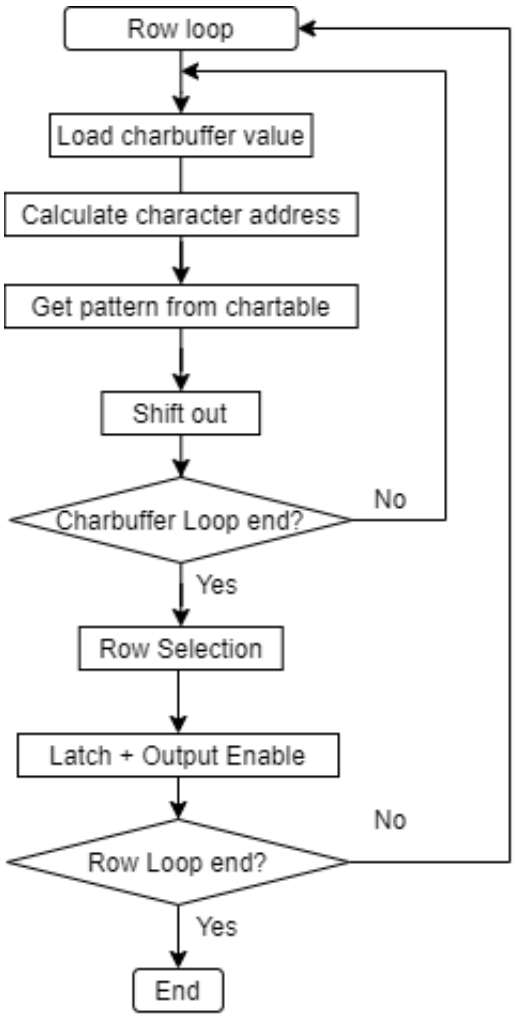

Figure 1: Screen Display

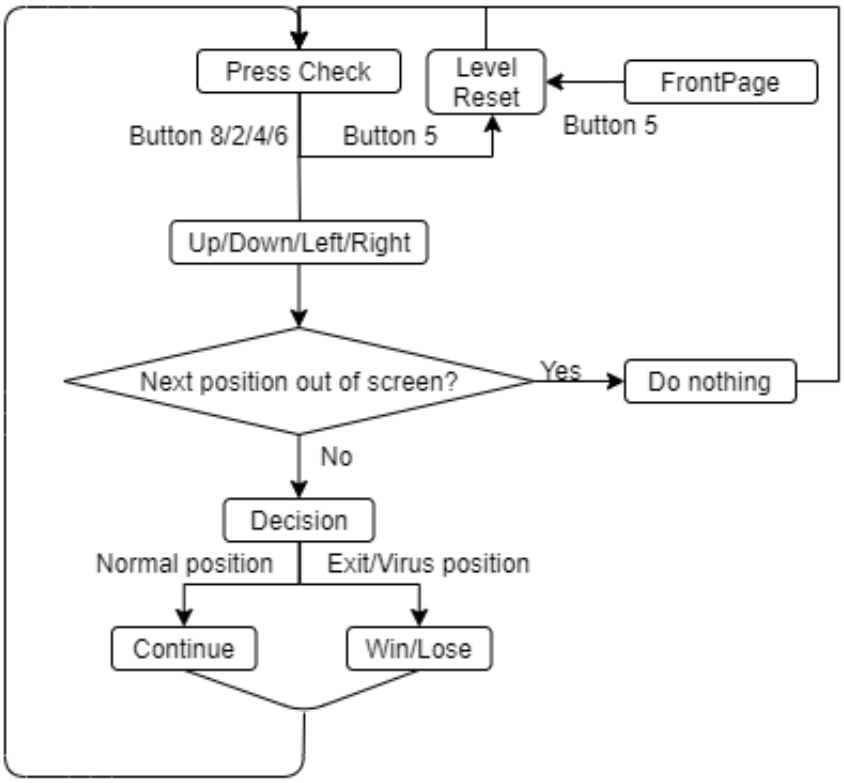

Figure 2: Game Strategy

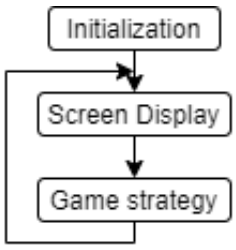

Figure 3: Put Together