COMP9032 Project Design Manual

Jingyun Shen z5119202

Contents

0.1 Introduction 2
0.2 Control Procedure 3
0.3 Modules 4
0.3.1 LCD Display 4
0.3.2 Keypad 5
0.3.3 Motor 6
0.3.4 Random Position of the Ball 7
0.3.5 Delay 7
0.4 Advantages 7

0.1 Introduction

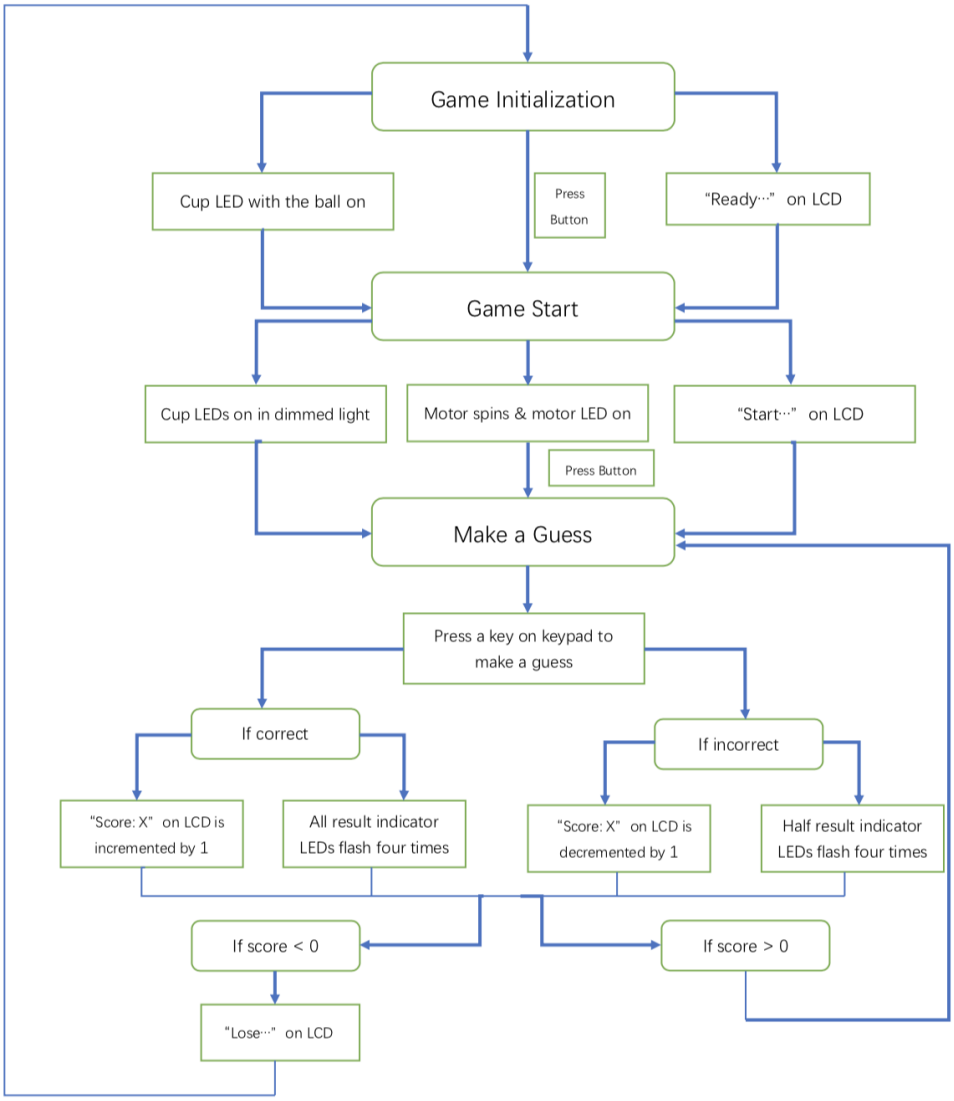
This project simulates an online game called Cup and Ball by using AVR Lab board. In this game, a ball is shuffled under three cups and the user can guess the position of the ball. For each guess, you gain one point if it is correct or lose one point if it is wrong.

In this system, we use the keypad, LCD, motor, LEDs and push buttons to simulate the game. The keypad is used for the user to input the number of cup which he guesses the ball is under. The user can press the number 1, 2 or 3 on the keypad, which indicates the ball is under LED0, LED1 or LED2, respectively. The LCD displays indicators in the game, such as the start of the game, and the scores the user gets. The operation of the motor is used to show the beginning of the game. The LEDs are in different use. LED0-2 are used to indicate three cups that may have balls hidden. LED3-6 are result indicator which indicates whether the user's guess is correct. LED7 is used to indicate the state of the motor. When the motor is running, LED7 is on, otherwise it is off. When the push button on the Lab board is pressed, the game starts.

The user can simulate the game multiple times. If the user loses all the scores, the simulation will be reset automatically and the game will restart.

0.2 Control Procedure

The whole control procedure of this simulation is demonstrated in the diagram.



1. Game Initialization
After the simulation system is turned on (i.e. the lab board is powered on), the system is initialized and the ball is with an arbitrarily cup. In this time, an indicator "Ready..." is displayed on the LCD. The cup LED with the ball is on and the other LEDs are off.

2. Game Start
When the oush button PB0 is pressed, the game starts and the ball is shuffled under the three cups. In this time, an indicator "Start..." is displayed on the LCD, the motor spins and LED7 is on. The three cup LEDs (LED0-2) are all on, but in dimmed light. The result indicator LEDs (LED3-6) are off.

3. Make a Guess
After the game starts, the user can start to make a guess by pressing the push button PB0 again. In this time, the motor stops and LED7 is off. The three cup LEDs remain dimmed. "Ball?" is displayed on LCD to indicate the user to choose the cup that he guesses the ball is under.

If the user guess that the bull is hidden in cup 1 (indicated by LED0), he can press "1" on the keypad. If the user guess that the bull is hidden in cup 2 (indicated by LED1), he can press "1" on the keypad. If the user guess that the bull is hidden in cup 3 (indicated by LED2), he can press "1" on the keypad. After making guesses, the user will have scores which is displayed on LCD as "Score: X".
If the guess is correct, the on the LCD is incremented by 1, and all the result indicator LEDs will flash four times.
If the guess is incorrect, the score on LCD is decremented, and half of the result indicator LEDs (LED3 on, LED4 off, LED5 on, LED6 off) will flash four times. If the score of the user is less than zero, the game restart automatically. "Lose." will be displayed on the LCD, and the system will go to the initial state. Moreover, if the first guess of a new game is incorrect, the score of the user is -1 and the game will restart. Thus, a user can get scores only if he win the first guess.

0.3 Modules

In this section, we are going to introduce the design of some important module in the simulation.

0.3.1 LCD Display

For LCD control, we use Port A and LCD data pins are connected to Port F. To send data to the LCD, we define a macro "do_lcd_data", and to send instructions to the LCD, we define a macro "do_lcd_command". Everytime we want to display something on the LCD or modify the display, we have to initialize it. To initialize the LCD, we use software initialization which is stored in a macro "reset_lcd". When we want to display some indicator, we first initialize the LCD and call the macro "do_lcd_data" with the characters we want to display one by one. When we display the scores that the user gets, we have to clear the LCD first, and then display the new score.

```
lcd_command:
        STORE LCD_DATA_PORT, r16
        rcall sleep_1ms
        lcd_set LCD_E
        rcall sleep_1ms
        lcd_clr LCD_E
        rcall sleep_1ms
        ret

lcd_data:
        STORE LCD_DATA_PORT, r16
        lcd_set LCD_RS
        rcall sleep_1ms
        lcd_set LCD_E
        rcall sleep_1ms
        lcd_clr LCD_E
        rcall sleep_1ms
        lcd_clr LCD_RS
        ret

.macro do_lcd_command
        ldi r16, @0
        rcall lcd_command
        rcall lcd_wait
.endmacro

.macro do_lcd_data
        mov r16, @0
        rcall lcd_data
        rcall lcd_wait
.endmacro

.macro reset_lcd
        ldi r16, low(RAMEND)
        out SPL, r16
        ldi r16, high(RAMEND)
        out SPH, r16

        ser r16
        STORE LCD_DATA_DDR, r16
        STORE LCD_CTRL_DDR, r16
        clr r16
        STORE LCD_DATA_PORT, r16
        STORE LCD_CTRL_PORT, r16


        do_lcd_command 0b00111000 ; 2x5x7
        rcall sleep_5ms
        do_lcd_command 0b00111000 ; 2x5x7
        rcall sleep_1ms
        do_lcd_command 0b00111000 ; 2x5x7
        do_lcd_command 0b00111000 ; 2x5x7
        do_lcd_command 0b00001000 ; display off
        do_lcd_command 0b00000001 ; clear display
        do_lcd_command 0b00000110 ; increment, no display shift
        do_lcd_command 0b00001110 ; Cursor on, bar, no blink
.endmacro
```

0.3.2 Keypad

To catch the input from the user on keypad, we have to scan each columns and rows on the keypad. We connects the keypad to Port L and set pin 7:4 as output and pin 3:0 as input. We define a column mask "cmask" and a row mask "rmask" to classify which key is pressed. In this project, we define a block call "keypad_scan" to achieve Continuous scanning. We scan the row first, and then scan each column. If we detect the change of mask, we locate the key and transfer the position of the key to the number it represents.

```
keypad_reset:
  ldi temp1, PORTFDIR ; PF7:4/PF3:0, out/in
  sts DDRL, temp1

keypad_scan:
        ldi cmask, INITCOLMASK ; initial column mask
        clr col ; initial column
colloop:
        cpi col, 4
        breq inital_result ; if all keys are scanned, repeat.
        sts PORTL, cmask ; otherwise, scan a column
        ldi temp1, 0xFF ; slow down the scan operation.
delay:
        dec temp1
        brne delay
        lds temp1, PINL ; read PORTF
        andi temp1, ROWMASK ; get the keypad output value
        cpi temp1, 0xF ; check if any row is low
        breq nextcol

        ; if yes, find which row is low
        ldi rmask, INITROWMASK ; initialize for row check
        clr row ;
        jmp rowloop

rowloop:
        cpi row, 4
        breq nextcol ; the row scan is over.
        mov temp2, temp1
        and temp2, rmask ; check un-masked bit
        breq convert ; if bit is clear, the key is pressed
        inc row ; else move to the next row
        lsl rmask
        jmp rowloop

nextcol: ; if row scan is over
        lsl cmask
        inc col ; increase column value
        jmp colloop ; go to the next column
```

0.3.3 Motor

To make the motor spin, we generate PWM waveform by Timer5. We connect the "mot" pin to LED7. When the motor spins, LED7 will also be on. To stop the motor, we only have to load a zero the the pin of LED7.

```
.macro motor_start
  ldi temp, 0b10000111
  sts DDRC, temp  |

  clr temp                  ; the value controls the PWM duty cycle
  sts OCR5AH, temp
  ldi temp, 0x4A
  sts OCR5AL, temp

  ; Set Timer0  to Phase Correct PWM mode  .
  ldi temp, (1 << CS50)
  sts TCCR5B, temp
  ldi temp, (1<< WGM50)|(1<<COM5A1)
  sts TCCR5A, temp
.endmacro

.macro motor_stop
  ldi temp, 0b00000111
  sts DDRC, temp
.endmacro
```

0.3.4 Random Position of the Ball

After each guess, the ball will be randomly put under another cup. To achieve this, we first define a macro called "random_cup". We use a register named "target_cup" to represent which cup the ball is currently under. The "random_cup" block basically moves the ball to the cup that next to its current cup. Then, we put this macro into the loop that judge whether the push button is pressed. In this way, the time that macro "random_cup" will be executed is depend on the time when the user spend to press the push button. Then, we can regard the position of the ball as random position.

```
.macro random_cup
clr r0
ldi r16, 2
mul target_cup, r16
mov target_cup, r0

cpi target_cup, 8
breq reset_cup
rjmp done_random

reset_cup:
ldi target_cup, 1

done_random:
.endmacro
```

```
if_guess:
  random_cup
  sbic PIND, 0
  rjmp if_guess
```

0.3.5 Delay

In this project, we use a common way to generate delay. Since we have to achieve the flash of LEDs, the minimal delay is set to be half second. First, we use a macro call "oneDelay", which executes the 8-cycle-instructions for 0xFF times. Then, in the macro "halfSecondDelay", it loops "oneDelay" 15 times. The whole time it spends on executing all these instructions is about half second.

```
.macro oneDelay                 ;delay the execution for 8 cyc
     ldi countL, low(loop_count)   ;set count=loop_count
     ldi countH, high(loop_count)
     clr iH                     ;clear i
     clr iL
loop:
     cp iL, countL  ;1 cyc     ;compare i with count
     cpc iH, countH  ;1 cyc
     brsh done          ;1-2 cyc   ;if i>= count,done
     adiw iH:iL, 1      ;2 cyc     ;if i<count, i+=1
     nop    ;1 cyc
     rjmp loop   ;2 cyc
done:
.endmacro

.macro halfSecondDelay   ; loop the oneDelay 30 times in order
                         ;to achieve half-second-delay
 clr r21             ; clear r21
loop1:
 cpi r21, 15         ; compare r21 to 15
 breq done1          ; if r21 == 30, done
 oneDelay            ; else do oneDelay
 inc r21             ; r21 ++
 rjmp loop1
done1:
.endmacro
```

0.4 Advantages

In this projract, we use AVR assembly programming languages and the Lab board to successfully simulate the Cup and Ball game. This simulation has the following advantages. 1. The whole process of the game runs smoothly. All the modules including LCD, LED, motor, push button and keypad work well.

2. We successfully achieve the random position of the ball by combining the time when the user spend on pressing the push button with the placement of the ball. This solution effectively achieves the expected function.
3. We solve the button debouncing problems by adding delays after the button is pressed. If the user presses the push button twice, different results will be brought out by his different pressing.
4. Last but not least, we add a lot of extra indicators that help the user to understand his situation in the game, which are not included in the project sheet. For example, if the user make an incorrect guess, half of the result indicator LEDs will flash. If the user loses all his scores, he will see "Lose..." on the LCD.