ENGG36 Final Project

Singles Ping-Pong Game

Ashley Garcia Cervantes

Project Overview

Using the LED lights in the dragon board, a light will bounce within each boundary, the player must hit the pushbuttons PH3 and PH0, for the left and right walls respectively, as soon as the light is visible in each boundary.

The game consists of three levels. The player will be able to access each level at the start of the program with the help of the DIP switches 1-3 (PH7-PH5). After each level the program will restart for the user to pick a level again, no need to reset dragon board unless the player wishes to quit playing. The score is kept and displayed in the seven-segment display at the end of the round. Score are the hits on each wall counted throughout the round, the maximum score is 5 per level but the player can keep playing until he/she losses, a score greater than 5 will not displayed. The program prevents cheating.

Game Specifications

How to play

The player must press pushbuttons PH0 and PH3 to bounce from the rightmost to the leftmost wall.[1]

Example code for rightmost wall (LD0):

```
        ;START!!
        bset    PortB, led0             ; LED 0 on;
        jsr     delay                   ; generate the desired delay
        ldaa    PTH
        brclr   PTH,$01,point           ; check if person pressed PH0 :), if they didn't, they lost
        lbra    lost

point   inc     score
        bclr    portB, led0             ; LED 0 off;
        jsr     delay                   ; generate delay again
```

Example code for leftmost wall (LD7):

```
keep1   bset    PortB, led7             ; LED 7 on;
        jsr     delay                   ; generate the desired delay

        brclr   PTH,$08,point2          ; check if person pressed PH3 :), if they didn't, they lost
        lbra    lost

point2  inc     score
        bclr    PortB, led7             ; LED 7 off;
        jsr     delay                   ; generate delay again
```

[1] It is necessary to flip the DIP switches 5 and 8 to logic 1 throughout the game in order for the program to run properly. As DIP switches 5-8 are interconnected with pushbuttons PH3-PH0.

Levels

The game consists of three levels, for each, a single light starts from the center and bounces out of the rightmost wall (LD0) first.

Level 1: Single Ping-Pong Ball (Slow)

A single light bounces from LD0 to LD7, generating a time delay after each light display with the **delay** subroutine.

```
; delay subroutine; use 2 loops to set the desired time delay
delay:        ldab   #$30                ; adjust the value to change blinking rate
delay1:       ldx    #$FFFF              ; generate a $FFFF long loop
delay2:       dbne   x,delay2            ; loop x amount of times
              dbne   b,delay1            ; loop throguh delay1 for b amount of times
              rts
```

Level 2: Single Ping-Pong Ball (Fast)

Similar to level 1, a single light bounces within the “walls”, but this time generating a faster time delay using the **delayPrime** subroutine.

```
; delay subroutine for level 2 and 3
delayPrime:     ldab    #$20                ; adjust the value to change blinking rate
delayPrime1:    ldx     #$FFFF              ; generate a $FFFF long loop
delayPrime2:    dbne    x,delayPrime2       ; loop x amount of times
                dbne    b,delayPrime1       ; loop throguh delay1 for b amount of times
                rts
```

Level 3: Doubles Ping-Pong Ball (Fast)

This level consists of two balls, and splits into four walls. Ball #1 bounces from LD0 to LD3 and ball #2 from LD4 to LD7.

The program starts like levels 1 and 2, but once the single ping pong ball bounces off the rightmost wall (LD0), the second ball begins from the leftward middle wall (LD4). The user uses push buttons PH0 and PH3 for balls #1 and #2 respectively, to bounce off each wall for the specified ball. This level is as fast as level 2 using the same **delayPrime** subroutine.

Score

The score increments after the ball bounces off each boundary. Since there are four walls on level 3, the score will increment only when the rightmost wall (LD0) and the leftmost wall (LD7) are hit.

Score is kept in the following format:

- If the player gets <5 wall hits, the number will be displayed in the seven-segment display.

- If the player gets >5 wall hits, the number will not be displayed, as the player can go as long as he/she desires. Once >5 wall hits are achieved, the player is ready for the next level.

Code for the **diplayScore** label

```
; displays scores up to 5!!
displayScore  ldaa   score

zero          cmpa   #$00                ; compare with digit 1
              bne    one
              movb   #$3F,PortB          ; show digit 1
              lbra   show

one           cmpa   #$01                ; compare with digit 1
              bne    two
              movb   #$06,PortB          ; show digit 1
              lbra   show

two           cmpa   #$02                ; compare with digit 2
              bne    three
              movb   #$5B,PortB          ; show digit 2
              lbra   show
three         cmpa   #$03                ; compare with digit 3
              bne    four
              movb   #$4F,PortB          ; show digit 3
              lbra   show
four          cmpa   #$04                ; compare with digit 4
              bne    five
              movb   #$66,PortB          ; show digit 4
              lbra   show
five          cmpa   #$05                ; compare with digit 5
              lbgt   continue            ; don't show if >5
              movb   #$6D,PortB          ; show digit 5
              lbra   show

show          movb   #$0E, PTP           ; turn on DISP1 (leftmost)
              jsr    delay               ; time delay for DISP1
              lbra   continue
```

Cheating Prevention

To prevent cheating the following code was implemented:

```
;Cheating Prevention
cheater       ldaa    #$00               ; display score of 0 for cheaters
              staa    score
              lbra    lost
```

A zero is displayed as the score and quits the round. The program jumps to label **cheater** when it senses the pushbutton PH0 or PH3 being pressed right before LED0 or LED7 is light up, respectively.

Example code for rightmost wall (LED0):

```
  bset    PortB, led1             ; LED 1 on;
  jsr     delay                   ; generate the desired delay
  bclr    PortB, led1             ; LED 1 off;
  jsr     delay                   ; generate delay again
  brclr   PTH,$01,cheater         ; check if person pressing PH0 :( CHEATER


  ;START!!
  bset    PortB, led0             ; LED 0 on;
  jsr     delay                   ; generate the desired delay
```

Example code for leftmost wall (LED7):

```
            bset    PortB, led6             ; LED 6 on;
            jsr     delay                   ; generate the desired delay
            bclr    PortB, led6             ; LED 6 off;
            jsr     delay                   ; generate delay again

            brset   PTH,$08,keep1           ; check if person pressing PH3 :( CHEATER
            lbra    cheater

keep1       bset    PortB, led7             ; LED 7 on;
            jsr     delay                   ; generate the desired delay
```

Cheating prevention is slightly different for level 3, as continuous pushbutton pressing is tested for the leftmost and rightmost wall of each half. Before LD7, LD4, LD3 and LD0 are on.

Code for right half - leftmost wall (LD3):

```
              bclr    PortB, led5           ; LED 5 off;
              jsr     delayPrime            ; generate delay again
              brset   PTH,$01,keepL3.1      ; check if person pressing PH0 :( CHEATER
              lbra    cheater

    keepL3.1  bset    PortB, led3           ; LED 3 on;
              bset    PortB, led6           ; LED 6 on;
```

Code for left half - rightmost wall (LD4):

```
cont2         bclr    PortB, led5           ; LED 5 off;
              bclr    PortB, led0           ; LED 0 off;
              jsr     delayPrime            ; generate delay again
              brset   PTH,$08,keepL3.4      ; check if person pressing PH3 :( CHEATER
              lbra    cheater

keepL3.4      bset    PortB, led4           ; LED 4 on;
              jsr     delayPrime            ; generate the desired delay
```