# Sumo: A Japanese Wrestling Simulator

Tarrin Rasmussen and Nathan Tipton

October 11, 2016

## Contents

# 1 Introduction

A two-player game simulating a sumo wresting match. The game will use an LED bar graph and two push buttons. Each player will control one of the two sumo wrestlers, trying to push the other out of the ring or to the edge of the bar graph. The wrestlers shove each other apart before each move and the player with the quickest reaction push his or her opponent closer to the edge of the ring. Player's speed is controlled by a dip switch.

# 2 Scope

This document will cover: hardware and software design for the Sumo game, testing processes for customer requirements, algorithm development and code.

# 3 Design Overview

## 3.1 Requirements

1. The display shall consist of a 10-LED bar graph mounted horizontally.
2. There shall be 2 buttons, each in the proximity of a different end of the bar graph. Player 1 uses the button on the left and player 2 uses the button on the right.
3. There shall be a DIP switch to configure the speed of each player. The speed Sn for player n shall be interpreted as a 2-bit binary number, one switch per bit.
4. The buttons shall be sampled at least every 5 ms (milliseconds).
5. After the system is reset, the two center LEDs of the bar graph shall flash at a rate of 2 Hz. This rate will be controlled using a timer. The LED on the left represents player 1 and the LED on the right represents player 2.
6. Each player must press their button to indicate their readiness to play. Once a player presses their button, their LED shall be lit solidly.
7. At some random time at least 1 second but no more than 2 seconds after (a) both players indicate their readiness to play or (b) a move concludes that does not end the game, the leftmost lid LED shall move one spot to the left and the rightmost lit LED shall move one spot to the right. This event starts the move.
8. After the move starts, each player races to press their button. As soon as a button is pressed, the corresponding players lit LED moves back to its prior position and a timer is started.

9. If the timer in (8) expires before the opponent presses their button (and moves their lit LED), the quicker players lit LED shall move again and be adjacent to their opponent's lit LED, Otherwise, the move is a draw.
10. If the result of this move is that the two lit LEDs are on the leftmost or rightmost side of the bar graph, the game is over and the 2 lit LEDs shall flash at a rate of 2 Hz until the system is reset.
11. The delay time in (8) shall be based on the players speed, Sn, and the number of contiguous drawn moves, d. If a player n is the first to press their button, the delay in milliseconds shall be $2^{-min(d,4)}(320 - 80Sn)$.

## 3.2 Dependencies

This design depends on:

1. 3.3 V Power Supply
2. Jumper wire
3. 10-LED bar graph
4. 2 push buttons
5. DIP switch
6. TIVA C EK-TM4C123GXL board
7. Breadboard

## 3.3 Theory of operation

### 3.3.1 Hardware

The game is displayed on a 10-LED bar graph. LEDs are controlled by the TIVA C microcontroller using Ports A and B. User input is obtained through 2 push buttons and a 4x1 DIP switch. All inputs are fed into the microcontroller. An external power supply powers the design.

### 3.3.2 Software

The game algorithm is modeled by the flowchart provided in Fig. 1.

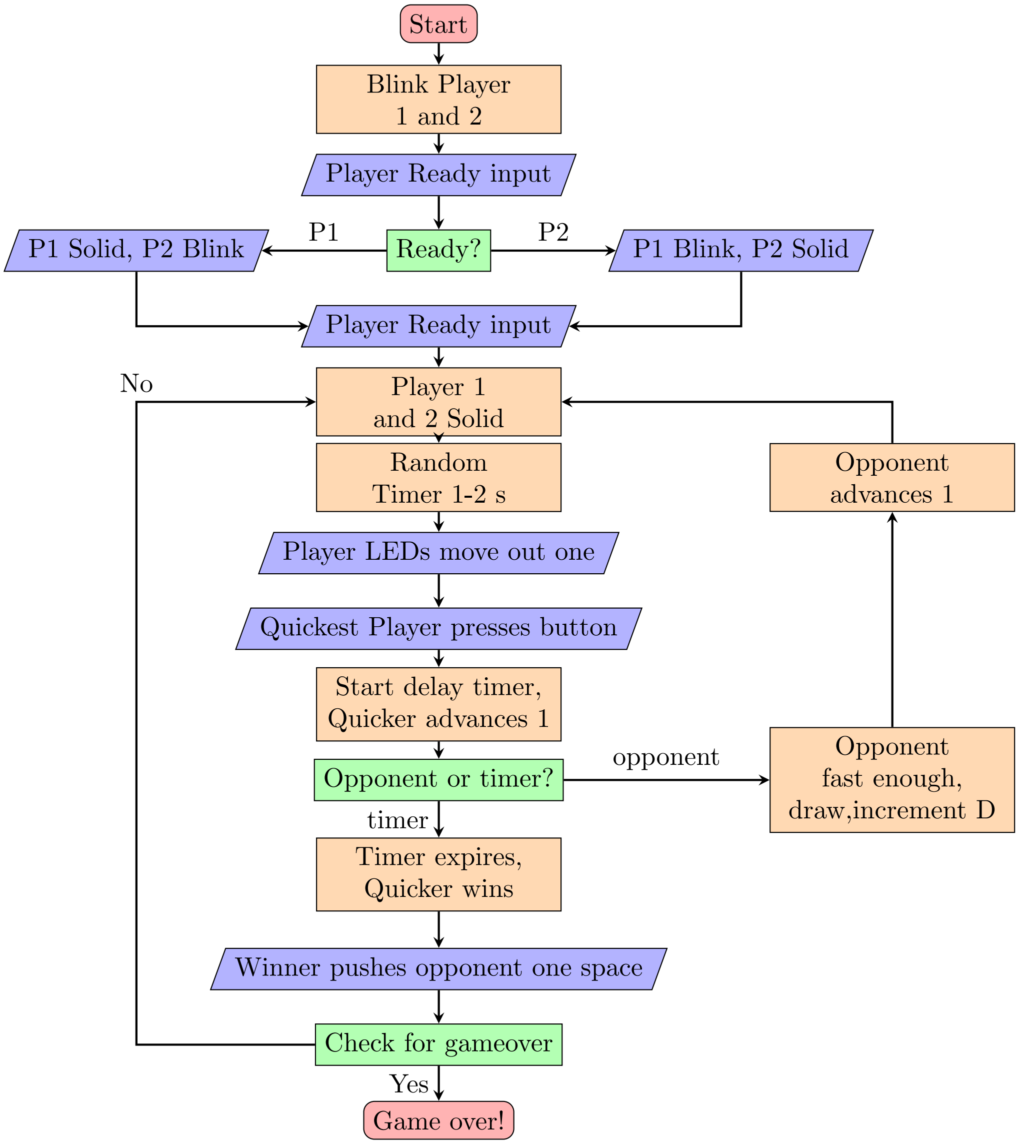

Figure 1: Algorithm Flowchart.

# 4 Design Details

## 4.1 Hardware

Pull up resistors were used for all LEDs, buttons, and switches. Current is sunk by the microcontroller, because the GPIO pins have a max of 8 mA drive strength. GPIO Port A [7:2] and Port B [3:0] were used for the LED bar graph, Port C [5:4] were used for the button inputs, and Port D [3:0] were used for the DIP switch. The resistor the bar was used to limit current through nine of the ten LEDs, and an additional external for the tenth. An external power supply was connected the breadboard to source the current for the LED bar graph.

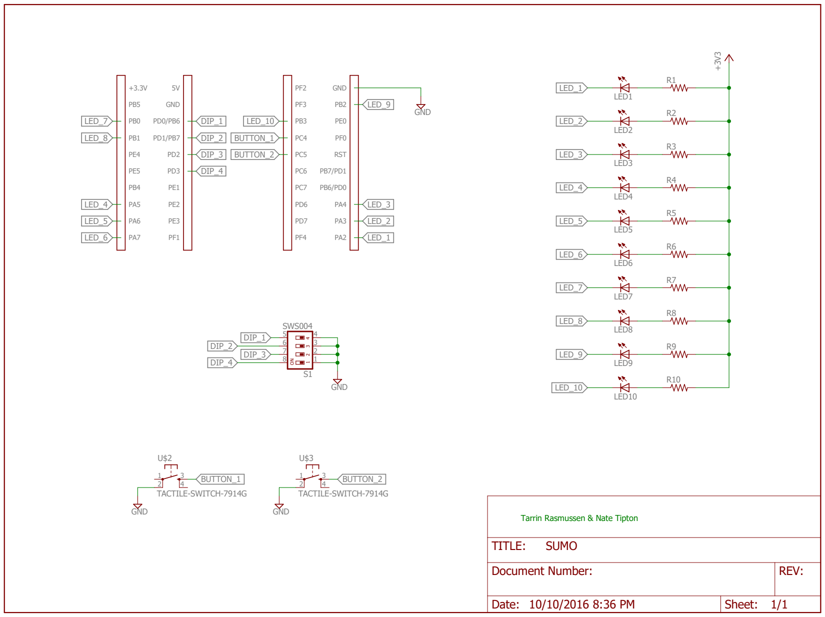

Figure 2: Schematic

## 4.2 Software

At reset, player speed are pulled from the 4 position DIP switch. These two bit player speeds, combined with the continuous draw count, are used to load delay times from a predefined lookup table in SRAM. Delay times are loaded into a timer that is enabled when the first button is pressed each round. The delay timer is sampled along with the second button to decide round outcome. The system clock is initialized at 16 MHz.

A random delay between 1-2 seconds is obtained by sampling Systick, which is periodic one second countdown. This sample is loaded every round when the second button is pressed. The sample value is subtracted from two seconds to generate a random delay in the specified interval.

# 5 Testing

Testing was done using an oscilloscope and logic analyzer. The delay timer was tested with conditions of zero as player speed and continuous draws. The calculated delay time was 320 ms. When measured on the logic analyzer, the delay time produced was 320 ms. Keil uVision debugger was used to test random values generated by systick. A breakpoint was placed in order to see the generated value for each iteration. Values were verified to be in the desired interval. Rigorous game play was used to test algorithm logic.

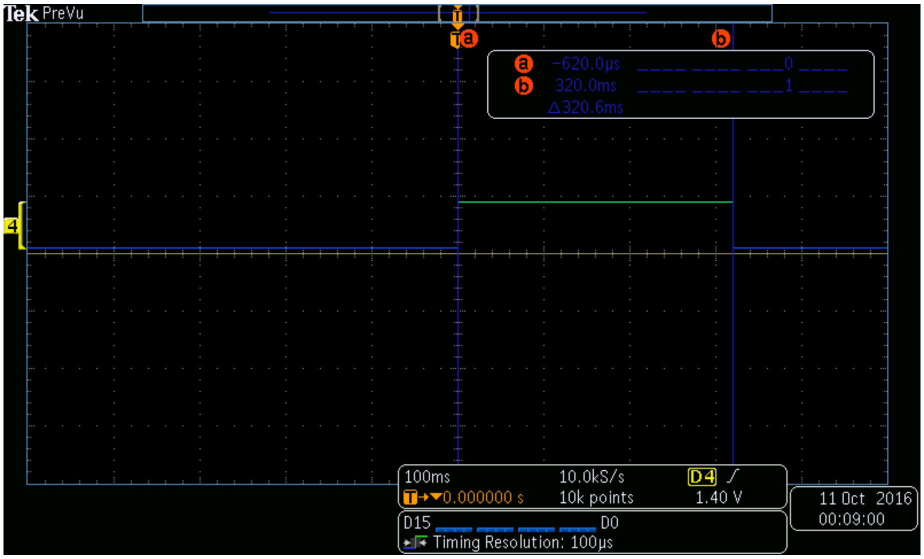

Figure 3: Delay Verification: Sn = 0, D = 0

# 6 Conclusion

All requirements were met when design was tested in the laboratory. The game design is fully functional. Design could be optimized with a cleaner hardware setup. Current setup makes it difficult to access the push buttons. The use of timers as opposed to delay loops was found to be more accurate and improved overall performance. Code optimization, such as the lookup table, was used to minimize runtime computations that would affect code and timing performance.

# 7 Appendix

## 7.1 Code

```
    THUMB
    AREA DATA, ALIGN=2
    ALIGN
    AREA |.text|, CODE, READONLY, ALIGN=2
    EXPORT Start


Start

        MOV32 R0, #0x400FE060 ;RCC Reset
        MOV32 R1, #0x078E3AD1
        STR R1, [R0]

    MOV32 R0, #0x40004000 ;Port A Base
    MOV32 R2, #0x400FE608 ;Clock Base

    ;Enable Clock
    LDR R1, [R2]
    ORR R1, #0x3F ;0b00111111
    STR R1, [R2]

    NOP
        NOP

    ;
    ; Setup Port A [7:2]
    ;

    ;Set Direction
    LDR R1, [R0, #0x400]
    ORR R1, #0xFC ;#0b11111100
    STR R1, [R0, #0x400]

    ;Disable Alternate
    LDR R1, [R0, #0x420]
    AND R1, #0x3 ;#0b00000011
    STR R1, [R0, #0x420]

    ;Set Drive Strength
    LDR R1, [R0, #0x508]
    ORR R1, #0xFC ;#0b11111100
    STR R1, [R0, #0x508]

    ;Pin Function
    LDR R1, [R0, #0x50C]
    ORR R1, #0xFC ;#0b11111100
    STR R1, [R0, #0x50C]

    ;Enable Pin
    LDR R1, [R0, #0x51C]
    ORR R1, #0xFC ;#0b11111100
    STR R1, [R0, #0x51C]

    ;Write to Pin
    LDR R1, [R0, #0x3F0]
    ORR R1, #0xFF ;#0b11111111
    STR R1, [R0, #0x3F0]


    ;
    ; Setup Port B [3:0]
    ;

    MOV32 R0, #0x40005000 ;Port B Base

    ;Set Direction
    LDR R1, [R0, #0x400]
    ORR R1, #0x0F ;#0b00001111
    STR R1, [R0, #0x400]
```

```
;Disable Alternate
LDR R1, [R0, #0x420]
AND R1, #0xF0 ;#0b11110000
STR R1, [R0, #0x420]

;Set Drive Strength
LDR R1, [R0, #0x508]
ORR R1, #0x0F ;#0b00001111
STR R1, [R0, #0x508]

;Pin Function
LDR R1, [R0, #0x50C]
MOV R1, #0xFF ;#0b11111111
STR R1, [R0, #0x50C]

;Enable Pin
LDR R1, [R0, #0x51C]
ORR R1, #0x0F ;#0b00001111
STR R1, [R0, #0x51C]

;Write to Pin
LDR R1, [R0, #0x3C]
MOV R1, #0x0F ;#0b00001111
STR R1, [R0, #0x3C]


;
; Setup Port C [4,5]
;

MOV32 R0, #0x40006000 ;Port C Base

;Set Direction
LDR R1, [R0, #0x400]
AND R1, #0xCF ;#0b11001111
STR R1, [R0, #0x400]

;Disable Alternate
LDR R1, [R0, #0x420]
AND R1, #0xCF ;#0b11001111
STR R1, [R0, #0x420]

;Set Drive Strength
LDR R1, [R0, #0x508]
ORR R1, #0x30 ;#0b00110000
STR R1, [R0, #0x508]

;Pin Function
LDR R1, [R0, #0x510]
ORR R1, #0x30 ;#0b00110000
STR R1, [R0, #0x510]

;Enable Pin
LDR R1, [R0, #0x51C]
ORR R1, #0x30 ;#0b00110000
STR R1, [R0, #0x51C]

;Write to Pin
;LDR R1, [R0, #0x3F0]
;ORR R1, #0xFF ;#0b11111111
;STR R1, [R0, #0x3F0]

            ;
; Setup Port D [3:0]
;

MOV32 R0, #0x40007000 ;Port D Base
```

```
;Set Direction
LDR R1, [R0, #0x400]
AND R1, #0xF0 ;#0b11110000
STR R1, [R0, #0x400]

;Disable Alternate
LDR R1, [R0, #0x420]
AND R1, #0xF0 ;#0b11110000
STR R1, [R0, #0x420]

;Set Drive Strength
LDR R1, [R0, #0x508]
ORR R1, #0x0F ;#0b00001111
STR R1, [R0, #0x508]

;Pin Function
LDR R1, [R0, #0x510]
ORR R1, #0x0F ;#0b00001111
STR R1, [R0, #0x510]

;Enable Pin
LDR R1, [R0, #0x51C]
ORR R1, #0x0F ;#0b00001111
STR R1, [R0, #0x51C]

;Write to Pin
;LDR R1, [R0, #0x3F0]
;ORR R1, #0xFF ;#0b11111111
;STR R1, [R0, #0x3F0]

;
; Counter
;

MOV32 R0, #0x40004000 ;Port A Base
MOV32 R1, #0x40005000 ;Port B Base
MOV32 R2, #0x40006000 ;Port C Base
MOV32 R3, #0x40007000 ;Port D Base
        MOV32 R4, #0xE000E000 ;Systick Base
        MOV32 R5, #0x40030000 ;Timer 0 Base (32−bit)
        MOV32 R6, #0xFFFFFFFF ;Blink output

        ;
; Setup Timer 0, A
;

        ;Enable Software
        MOV32 R8, #0x400FE000
        LDR R7, [R8, #0x604]
        ORR R7, #1
        STR R7, [R8, #0x604]

        NOP
        NOP

        ;Disable Timer
        LDR R7, [R5, #0xC]
        AND R7, #0xFFFFFFFE
        STR R7, [R5, #0xC]

        ;Timer Config (16/32/64)
        MOV R7, #0x0
        STR R7, [R5]

        ;Timer Mode
        MOV R7, #0x2
        STR R7, [R5, #0x4]
        ;Start Value
        MOV32 R7, #4000000
        ;MOV32 R7, #0x003A9800
        STR R7, [R5, #0x28]

        ;Enable Timer
        LDR R7, [R5, #0xC]
        ORR R7, #0x1
        STR R7, [R5, #0xC]

        ;
; SUMO Game
;

        MOV32 R8, #0x20002000
        LDR R7, [R3, #0xC] ;Save speeds to SRAM
        STR R7, [R8]
        LDR R7, [R3, #0x30]
        LSR R7, #2
        STR R7, [R8, #4]

        ;Lookup Table
        MOV32 R8, #0x20002008
        MOV32 R9, #5120000 ;Sn = 0, D = 0 (320ms) Sn = 0
        STR R9, [R8, #0]
        MOV32 R9, #2560000 ;Sn = 0, D = 1 (160ms)
        STR R9, [R8, #4]
        MOV32 R9, #1280000 ;Sn = 0, D = 2 (80ms)
        STR R9, [R8, #8]
        MOV32 R9, #640000 ;Sn = 0, D = 3 (40ms)
        STR R9, [R8, #12]
        MOV32 R9, #320000 ;Sn = 0, D = 4 (20ms)
        STR R9, [R8, #16]

        MOV32 R8, #0x2000201C
        MOV32 R9, #3840000 ;Sn = 1, D = 0 (240ms) Sn = 1
        STR R9, [R8, #0]
        MOV32 R9, #1920000 ;Sn = 1, D = 1 (120ms)
        STR R9, [R8, #4]
        MOV32 R9, #960000 ;Sn = 1, D = 2 (60ms)
        STR R9, [R8, #8]
        MOV32 R9, #480000 ;Sn = 1, D = 3 (30ms)
        STR R9, [R8, #12]
        MOV32 R9, #240000 ;Sn = 1, D = 4 (15ms)
        STR R9, [R8, #16]

        MOV32 R8, #0x20002030
        MOV32 R9, #2560000 ;Sn = 2, D = 0 (160ms) Sn = 2
        STR R9, [R8, #0]
        MOV32 R9, #1280000 ;Sn = 2, D = 1 (80ms)
        STR R9, [R8, #4]
        MOV32 R9, #640000 ;Sn = 2, D = 2 (40ms)
        STR R9, [R8, #8]
        MOV32 R9, #320000 ;Sn = 2, D = 3 (20ms)
        STR R9, [R8, #12]
        MOV32 R9, #160000 ;Sn = 2, D = 4 (10ms)
        STR R9, [R8, #16]

        MOV32 R8, #0x20002044
        MOV32 R9, #1280000 ;Sn = 3, D = 0 (80ms) Sn = 3
        STR R9, [R8, #0]
        MOV32 R9, #640000 ;Sn = 3, D = 1 (40ms)
        STR R9, [R8, #4]
        MOV32 R9, #320000 ;Sn = 3, D = 2 (20ms)
        STR R9, [R8, #8]
        MOV32 R9, #160000 ;Sn = 3, D = 3 (10ms)
        STR R9, [R8, #12]
        MOV32 R9, #80000 ;Sn = 3, D = 4 (5ms)
```

```
                    STR R9, [R8, #16]

                    MOV R9, #1 ;Reset Flag

init MVN R6, R6 ;Blink Port A
                    STR R6, [R0, #0x300]

wait LDR R7, [R2, #0xC0] ;Checks Buttons
          CMP R7, #0x30
                    BNE systick

                    LDR R7, [R5, #0x1C] ;Check Timer 0
                    CMP R7, #0
                    BEQ wait

                    STR R9, [R5, #0x24] ;Reset Timer

                    B init

systick MOV32 R8, #16000000 ;Set Random Timer
                    STR R8, [R4, #0x14]
                    MOV R8, #1
                    STR R8, [R4, #0x10]

wait2 LDR R8, [R2, #0xC0] ;Checks Buttons
          CMP R8, #0x20
                    BEQ b2
                    CMP R8, #0x10
                    BEQ b1
                    B wait2


b1
                    MOV R7, #0x0
                    STR R7, [R0,#0x100]
                    MVN R6, R6 ;Blink Port A
                    STR R6, [R0, #0x200]

blink1

                    LDR R7, [R2, #0xC0] ;Checks Buttons
          CMP R7, #0x20
                    BEQ play

                    LDR R7, [R5, #0x1C] ;Check Timer 0
                    CMP R7, #0
                    BEQ blink1

                    STR R9, [R5, #0x24] ;Reset Timer

                    B b1

b2
                    MOV R7, #0x0
                    STR R7, [R0,#0x200]
                    MVN R6, R6 ;Blink Port A
                    STR R6, [R0, #0x100]

blink2

                    LDR R7, [R2, #0xC0] ;Checks Buttons
          CMP R7, #0x10
                    BEQ play

                    LDR R7, [R5, #0x1C] ;Check Timer 0
                    CMP R7, #0
                    BEQ blink2
```

```
                STR R9, [R5, #0x24] ;Reset Timer

                B b2

                ;Disable Timer
                LDR R7, [R5, #0xC]
                AND R7, #0xFFFFFFFE
                STR R7, [R5, #0xC]

                ;Timer Config (16/32/64)
                MOV R7, #0x0
                STR R7, [R5]

                ;Timer Mode
                MOV R7, #0x1
                STR R7, [R5, #0x4]

                MOV R6, #0x0 ;Counter

play
                MOV R7, #0x0 ;Center two solid
                STR R7, [R0,#0x300]

                MOV R9, R0 ;P1 Base
                MOV R10, #0x100 ;P1 Mask
                MOV R11, R0 ;P2 Base
                MOV R12, #0x200 ;P2 Mask

loop
                CMP R6,#5
                MOVEQ R6,#4
                MOV R7, #0xFC ;Reset Port A lights
                STR R7, [R0, #0x3F0]
                MOV R7, #0xF ;Reset Port B lights
                STR R7, [R1, #0x3C]
                MOV R8, #0
                STR R8, [R9, R10] ;Turn on player 1
                STR R8, [R11, R12] ;Turn on player 2

                MOV32 R4, #0xE000E000 ;Systick Base
                LDR R8,[R4,#0x18] ;Random time between 1−2 sec
                MOV32 R7,#32000000
                SUB R8,R7,R8
                ;LSR R8, #1

                MOV R7, #1 ;Reset Flag
                STR R7, [R5, #0x24] ;Reset Timer

                ;Start Value
                STR R8, [R5, #0x28]

                ;Enable Timer
                LDR R7, [R5, #0xC]
                ORR R7, #0x1
                STR R7, [R5, #0xC]

random
                LDR R7, [R5, #0x1C] ;Check Timer 0
                CMP R7, #1
                BNE random

begin MOV32 R4, #0x20002000
                MOV R7, #0xFC ;Reset Port A lights
                STR R7, [R0, #0x3F0]
                MOV R7, #0xF ;Reset Port B lights
                STR R7, [R1, #0x3C]
                BL p2l
                BL p1r
                MOV R8, #0
                STR R8, [R9, R10] ;Turn on player 1
                STR R8, [R11, R12] ;Turn on player 2

                MOV R7, #1 ;Reset Flag
                STR R7, [R5, #0x24] ;Reset Timer

check LDR R7, [R2, #0xC0] ;Checks Buttons
        CMP R7, #0x30
                BEQ check
                CMP R7, #0x0
                BEQ check

                CMP R7, #0x10 ;Button one
                BEQ Player1

                CMP R7, #0x20 ;Button two
                BEQ Player2

Player1 LDR R8, [R4]
                MOV R4, #0x14 ;Get table base
                MUL R8, R4
                MOV32 R4, #0x20002008
                ADD R4, R8

                LDR R8, [R4, R6, LSL #2]

                ;Start Value
                STR R8, [R5, #0x28]

                ;Enable Timer
                LDR R7, [R5, #0xC]
                ORR R7, #0x1
                STR R7, [R5, #0xC]

checkp2
                LDR R7, [R5, #0x1C] ;Check Timer 0
                CMP R7, #1
                BEQ win
                LDR R7, [R2, #0xC0] ;Checks Buttons
        CMP R7, #0x20
                BNE checkp2


draw
                ADD R6,#0x1
                BL p1l
                BL p2r
                B loop

win
                MOV R6,#0x0
                BL p1l
                BL p1l
                B loop

Player2 LDR R8, [R4, #0x4]
                MOV R4, #0x14 ;Get table base
                MUL R8, R4
                MOV32 R4, #0x20002008
                ADD R4, R8

                LDR R8, [R4, R6, LSL #2]

                ;Start Value
                STR R8, [R5, #0x28]

                ;Enable Timer
                LDR R7, [R5, #0xC]
                ORR R7, #0x1
                STR R7, [R5, #0xC]

checkp1
                LDR R7, [R5, #0x1C] ;Check Timer 0
                CMP R7, #1
                BEQ win2
                LDR R7, [R2, #0xC0] ;Checks Buttons
        CMP R7, #0x10
                BNE checkp1


draw2
                ADD R6,#0x1
                BL p2r
                BL p1l
                B loop

win2
                MOV R6,#0x0
                BL p2r
                BL p2r
                B loop

p1r  CMP R9, R0 ;Player 1 Shift Right
                BNE b1r
                CMP R10, #16
                BEQ gameover1
                LSR R10, #1
                BX LR
b1r  CMP R10, #4
                MOVEQ R9, R0
                MOVEQ R10, #0x200
                LSRNE R10, #1
                BX LR

p2r  CMP R11, R0 ;Player 2 Shift Right
                BNE b2r
                LSR R12, #1
                BX LR
b2r  CMP R12, #4
                MOVEQ R11, R0
                MOVEQ R12, #0x200
                LSRNE R12, #1
                BX LR

p1l  CMP R9, R0 ;Player 1 Shift Left
                BNE b1l
                CMP R10, #0x200
                MOVEQ R9, R1
                MOVEQ R10, #4
                LSLNE R10, #1
                BX LR
b1l  LSL R10, #1
                BX LR

p2l  CMP R11, R0 ;Player 2 Shift Left
                BNE b2l
                CMP R12, #0x200
                MOVEQ R11, R1
                MOVEQ R12, #4
                LSLNE R12, #1
                BX LR
b2l  CMP R12, #0x20
                BEQ gameover2
                LSL R12, #1
                BX LR
```

```
gameover1
                ;Disable Timer
                LDR R7, [R5, #0xC]
                AND R7, #0xFFFFFFFE
                STR R7, [R5, #0xC]

                ;Timer Config (16/32/64)
                MOV R7, #0x0
                STR R7, [R5]

                ;Timer Mode
                MOV R7, #0x2
                STR R7, [R5, #0x4]

                ;Start Value
                MOV32 R7, #4000000
                STR R7, [R5, #0x28]

                ;Enable Timer
                LDR R7, [R5, #0xC]
                ORR R7, #0x1
                STR R7, [R5, #0xC]

                MOV32 R6, #0xFFFFFFFF ;Blink output

go1  MVN R6, R6 ;Blink Port A
                STR R6, [R0, #0x30]
gowait1
                LDR R7, [R5, #0x1C] ;Check Timer 0
                CMP R7, #0
                BEQ gowait1

                MOV R9, #1
                STR R9, [R5, #0x24] ;Reset Timer

                B go1

gameover2
                ;Disable Timer
                LDR R7, [R5, #0xC]
                AND R7, #0xFFFFFFFE
                STR R7, [R5, #0xC]

                ;Timer Config (16/32/64)
                MOV R7, #0x0
                STR R7, [R5]

                ;Timer Mode
                MOV R7, #0x2
                STR R7, [R5, #0x4]

                ;Start Value
                MOV32 R7, #4000000
                STR R7, [R5, #0x28]

                ;Enable Timer
                LDR R7, [R5, #0xC]
                ORR R7, #0x1
                STR R7, [R5, #0xC]

                MOV32 R6, #0xFFFFFFFF ;Blink output

go2  MVN R6, R6 ;Blink Port B
                STR R6, [R1, #0x30]
gowait2
                LDR R7, [R5, #0x1C] ;Check Timer 0
                CMP R7, #0
```

```
        BEQ gowait2

        MOV R9, #1
        STR R9, [R5, #0x24] ;Reset Timer

        B go2


    B Start

    ALIGN
    END
```