Final Project Report

Code Viewing

common parameters

這些是在各個`file`中都有出現以及其代表的意思

```
parameter start_scene = 4'b0001;
parameter choose_scene = 4'b0010;
parameter fight_scene = 4'b0011;
parameter win_scene = 4'b0100;
parameter poke_1 = 8'd1;
parameter poke_2 = 8'd2;
parameter poke_3 = 8'd3;
parameter poke_4 = 8'd4;
parameter poke_5 = 8'd5;
parameter poke_6 = 8'd6;
parameter poke_7 = 8'd7;
parameter poke_8 = 8'd8;

parameter [6-1:0] fight_state_menu = 6'd1;
parameter [6-1:0] fight_state_choosing_skill = 6'd2;
parameter [6-1:0] fight_state_animation_p1 = 6'd3; // p1 attack
parameter [6-1:0] fight_state_animation_p2 = 6'd4; // p2 attack
parameter [6-1:0] fight_state_hpReducing_p1 = 6'd5; // p1 reducing hp
parameter [6-1:0] fight_state_hpReducing_p2 = 6'd6; // p2 reducing hp
parameter [6-1:0] fight_state_p1_win = 6'd7;        // one of the player die
parameter [6-1:0] fight_state_p2_win = 6'd8;

parameter [4-1:0] option_state_1 = 4'd1;
parameter [4-1:0] option_state_2 = 4'd2;
parameter [4-1:0] option_state_3 = 4'd3;
parameter [4-1:0] option_state_4 = 4'd4;
```

Top and others

top.v

vga.v

state_control.v

OnePulse.v

clk_divisor.v

counter.v

pxiel_gen.v

scene

start_scene.v

choose_scene.v

fight_scene.v

win_scene.v

image_load_modules.v

data_control

choose_data_control.v

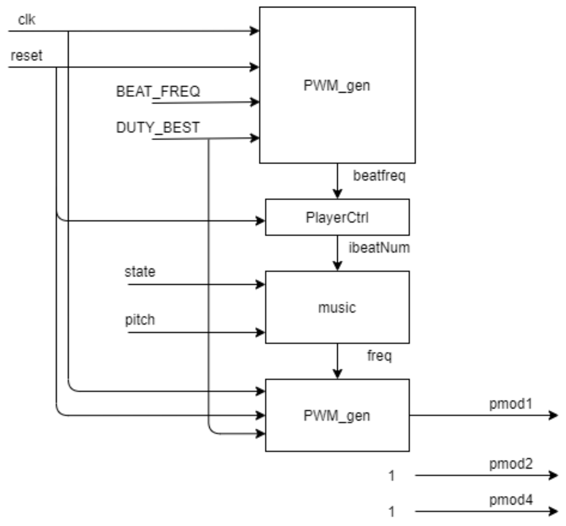
fight_data_control.v

keyboard

Keyboard_Decoder.v

music

music_top.v



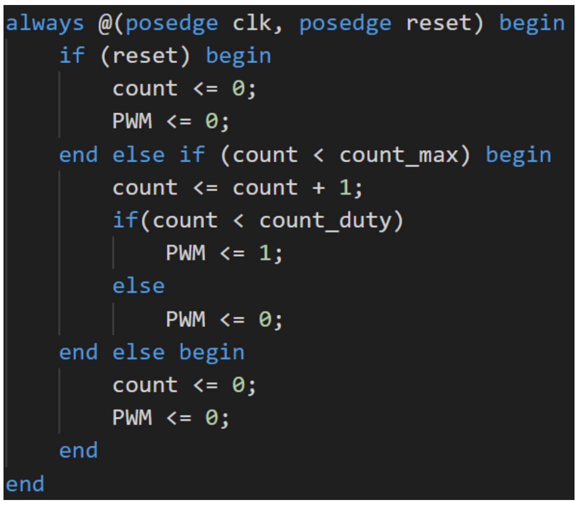
```
parameter BEAT_FREQ = 32'd12;   //one beat=1/12sec
parameter DUTY_BEST = 10'd512;  //duty cycle=50%
```

```
assign pmod_2 = 1'd1;    //no gain(6dB)
assign pmod_4 = 1'd1;    //turn-on
```

PWM_gen.v

```
wire [31:0] count_max = 100_000_000 / freq;
wire [31:0] count_duty = count_max * duty / 1024;
```

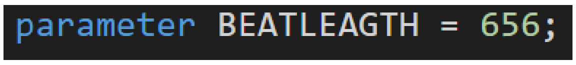
使用 count 產生PWM週期為



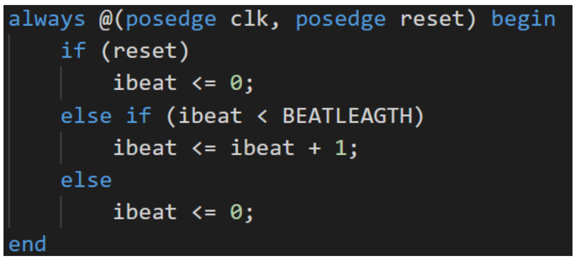
```verilog
always @(posedge clk, posedge reset) begin
    if (reset) begin
        count <= 0;
        PWM <= 0;
    end else if (count < count_max) begin
        count <= count + 1;
        if(count < count_duty)
            PWM <= 1;
        else
            PWM <= 0;
    end else begin
        count <= 0;
        PWM <= 0;
    end
end
```

count_max，1的長度為count_duty。

PlayerCtrl.v



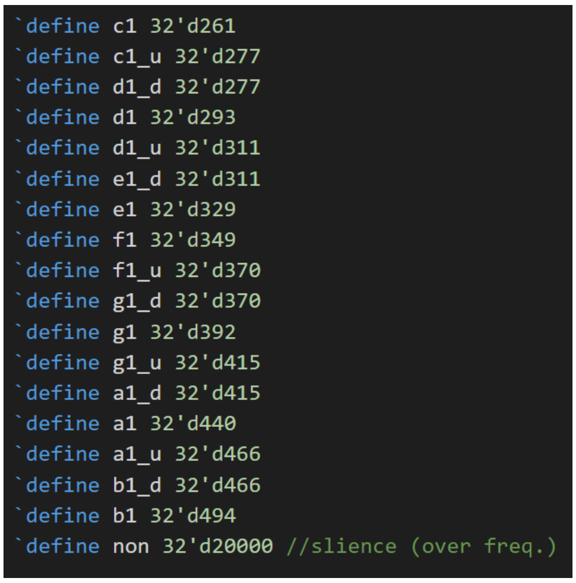
```verilog
parameter BEATLEAGTH = 656;
```



```verilog
always @(posedge clk, posedge reset) begin
    if (reset)
        ibeat <= 0;
    else if (ibeat < BEATLEAGTH)
        ibeat <= ibeat + 1;
    else
        ibeat <= 0;
end
```

從0開始算每一個音符，算到最後一個後，從頭開始循環。

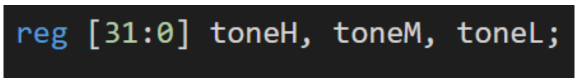
Music.v



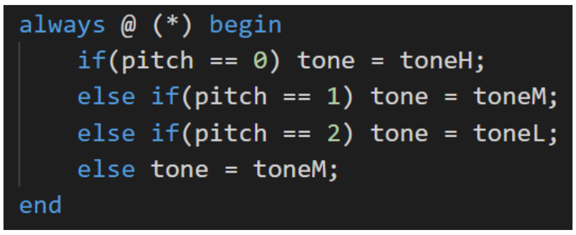
```verilog
`define c1 32'd261
`define c1_u 32'd277
`define d1_d 32'd277
`define d1 32'd293
`define d1_u 32'd311
`define e1_d 32'd311
`define e1 32'd329
`define f1 32'd349
`define f1_u 32'd370
`define g1_d 32'd370
`define g1 32'd392
`define g1_u 32'd415
`define a1_d 32'd415
`define a1 32'd440
`define a1_u 32'd466
`define b1_d 32'd466
`define b1 32'd494
`define non 32'd20000 //slience (over freq.)
```

先define在中央Do到Si的頻率，因為八度音是差兩倍頻率，所以所有音都可以用這些音表達。

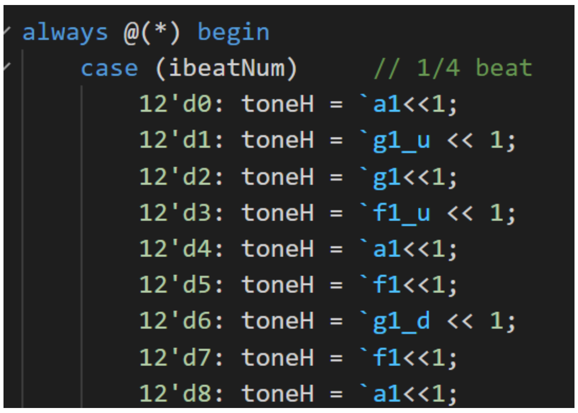
分成High，Mid，Low三個聲部。



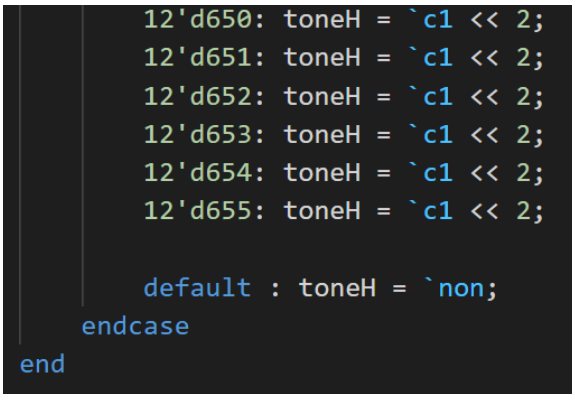
```
reg [31:0] toneH, toneM, toneL;
```



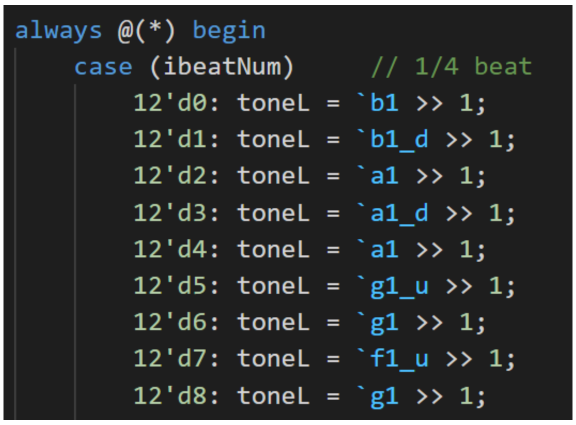
```
always @ (*) begin
    if(pitch == 0) tone = toneH;
    else if(pitch == 1) tone = toneM;
    else if(pitch == 2) tone = toneL;
    else tone = toneM;
end
```



```
always @(*) begin
    case (ibeatNum)     // 1/4 beat
        12'd0: toneH = `a1<<1;
        12'd1: toneH = `g1_u << 1;
        12'd2: toneH = `g1<<1;
        12'd3: toneH = `f1_u << 1;
        12'd4: toneH = `a1<<1;
        12'd5: toneH = `f1<<1;
        12'd6: toneH = `g1_d << 1;
        12'd7: toneH = `f1<<1;
        12'd8: toneH = `a1<<1;
```

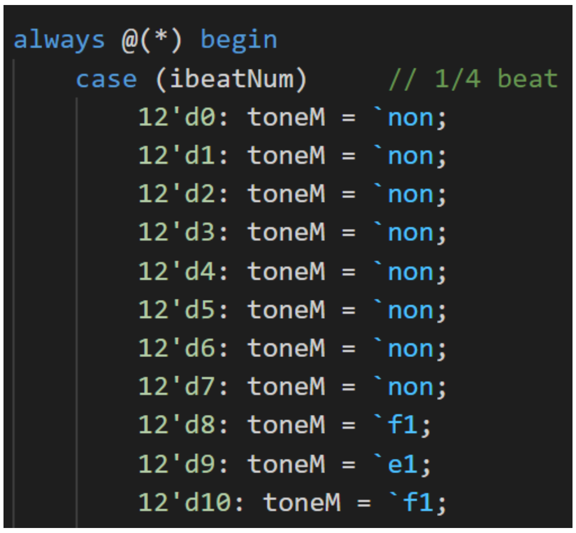


```
        12'd650: toneH = `c1 << 2;
        12'd651: toneH = `c1 << 2;
        12'd652: toneH = `c1 << 2;
        12'd653: toneH = `c1 << 2;
        12'd654: toneH = `c1 << 2;
        12'd655: toneH = `c1 << 2;

        default : toneH = `non;
    endcase
end
```

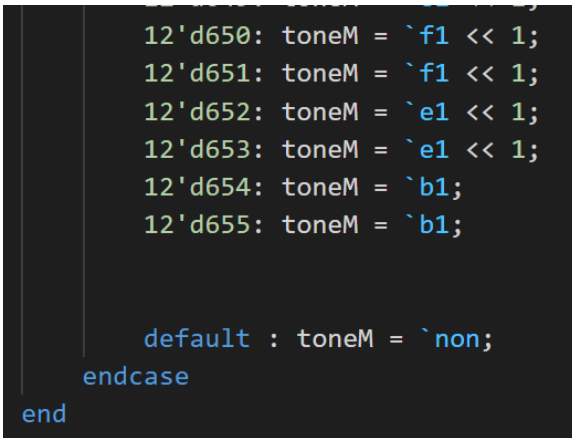


```
always @(*) begin
    case (ibeatNum)     // 1/4 beat
        12'd0: toneL = `b1 >> 1;
        12'd1: toneL = `b1_d >> 1;
        12'd2: toneL = `a1 >> 1;
        12'd3: toneL = `a1_d >> 1;
        12'd4: toneL = `a1 >> 1;
        12'd5: toneL = `g1_u >> 1;
        12'd6: toneL = `g1 >> 1;
        12'd7: toneL = `f1_u >> 1;
        12'd8: toneL = `g1 >> 1;
```

依靠pitch判斷現在的output是哪個聲部。

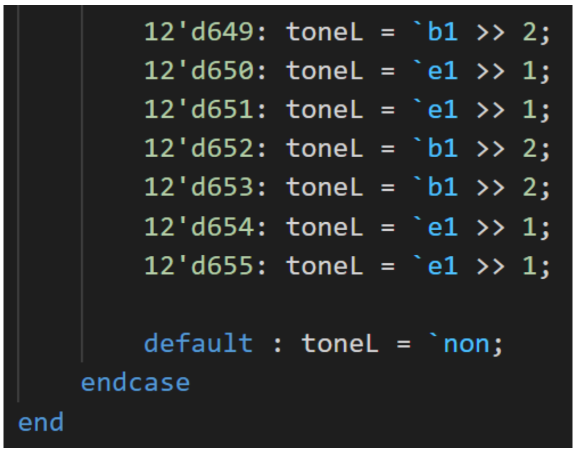


```
always @(*) begin
    case (ibeatNum)     // 1/4 beat
        12'd0: toneM = `non;
        12'd1: toneM = `non;
        12'd2: toneM = `non;
        12'd3: toneM = `non;
        12'd4: toneM = `non;
        12'd5: toneM = `non;
        12'd6: toneM = `non;
        12'd7: toneM = `non;
        12'd8: toneM = `f1;
        12'd9: toneM = `e1;
        12'd10: toneM = `f1;
```

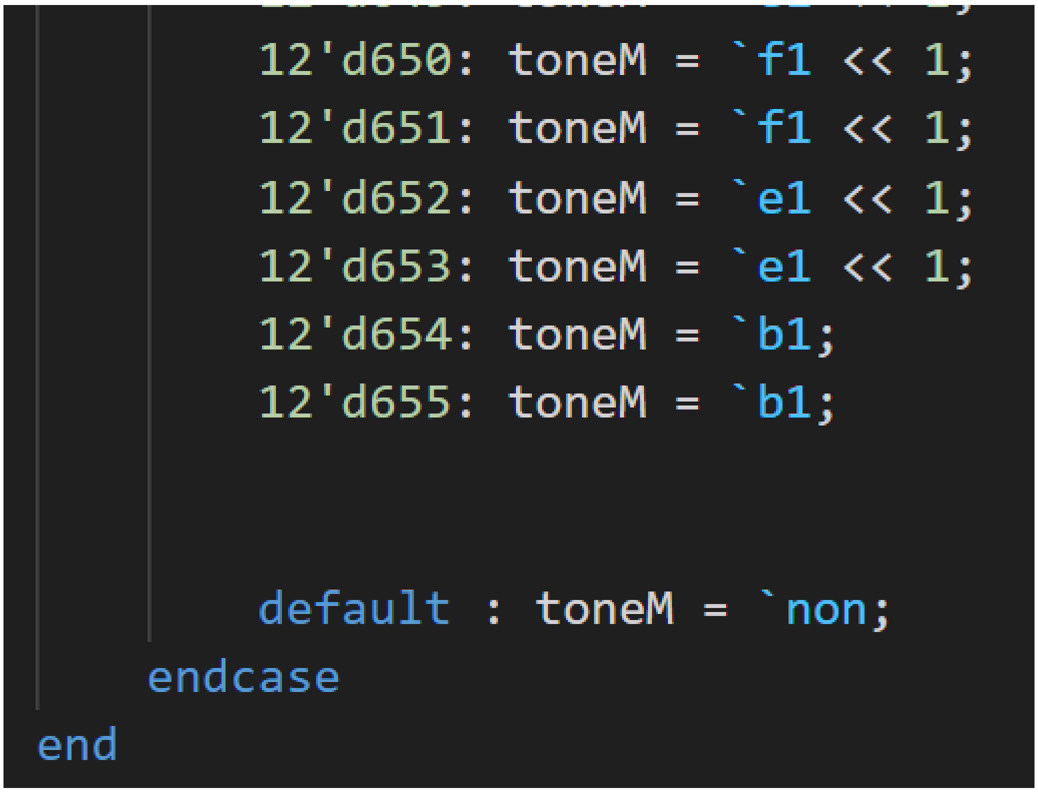


```
        12'd650: toneM = `f1 << 1;
        12'd651: toneM = `f1 << 1;
        12'd652: toneM = `e1 << 1;
        12'd653: toneM = `e1 << 1;
        12'd654: toneM = `b1;
        12'd655: toneM = `b1;


        default : toneM = `non;
    endcase
end
```

填入三聲部的所有音符，用ibeatNum決定現在是甚麼音符。



```verilog
            12'd649: toneL = `b1 >> 2;
            12'd650: toneL = `e1 >> 1;
            12'd651: toneL = `e1 >> 1;
            12'd652: toneL = `b1 >> 2;
            12'd653: toneL = `b1 >> 2;
            12'd654: toneL = `e1 >> 1;
            12'd655: toneL = `e1 >> 1;

            default : toneL = `non;
        endcase
end
```



```verilog
            12'd650: toneM = `f1 << 1;
            12'd651: toneM = `f1 << 1;
            12'd652: toneM = `e1 << 1;
            12'd653: toneM = `e1 << 1;
            12'd654: toneM = `b1;
            12'd655: toneM = `b1;


            default : toneM = `non;
        endcase
end
```