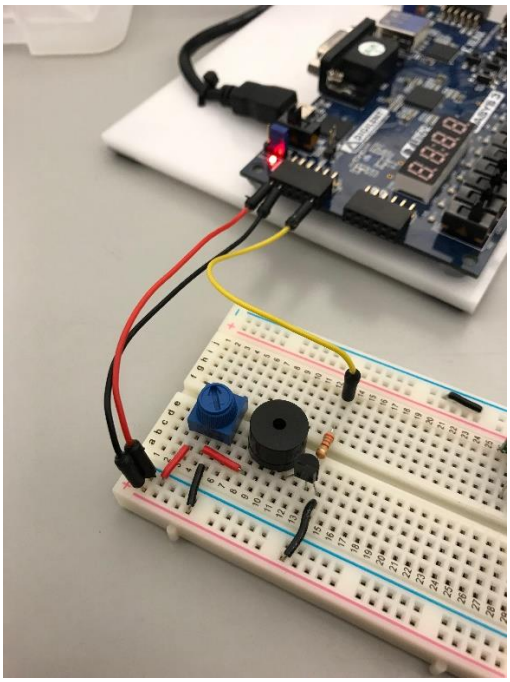
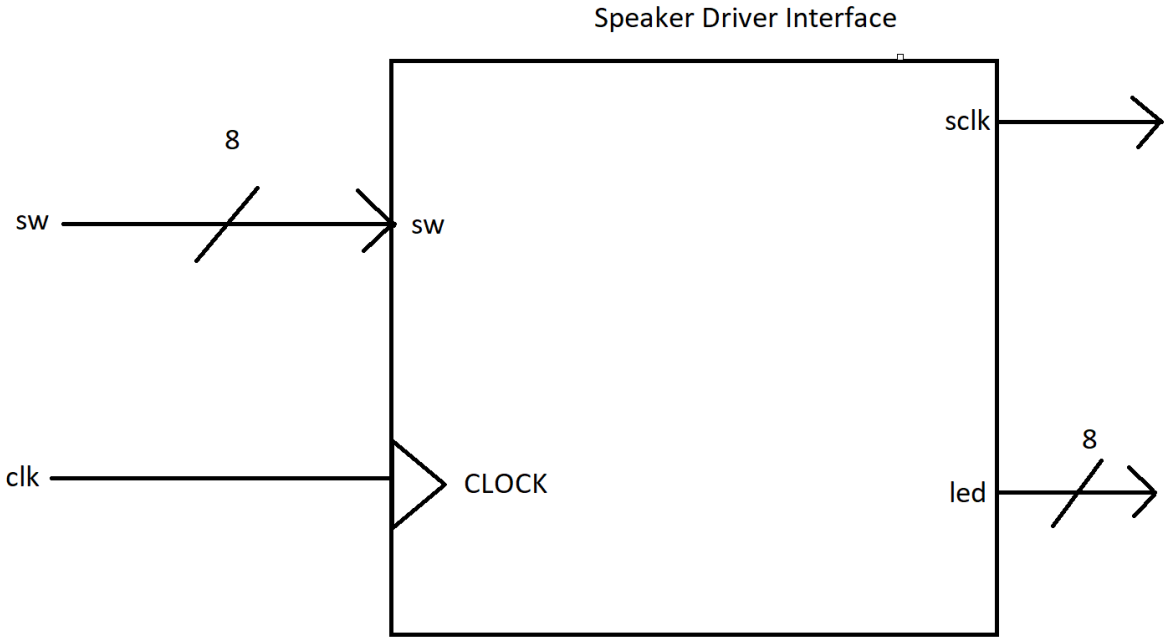
CPE 233: Peripheral assignment 1

Prof. Bridget Benson

Luis Gomez, Jared Rocha



BBD

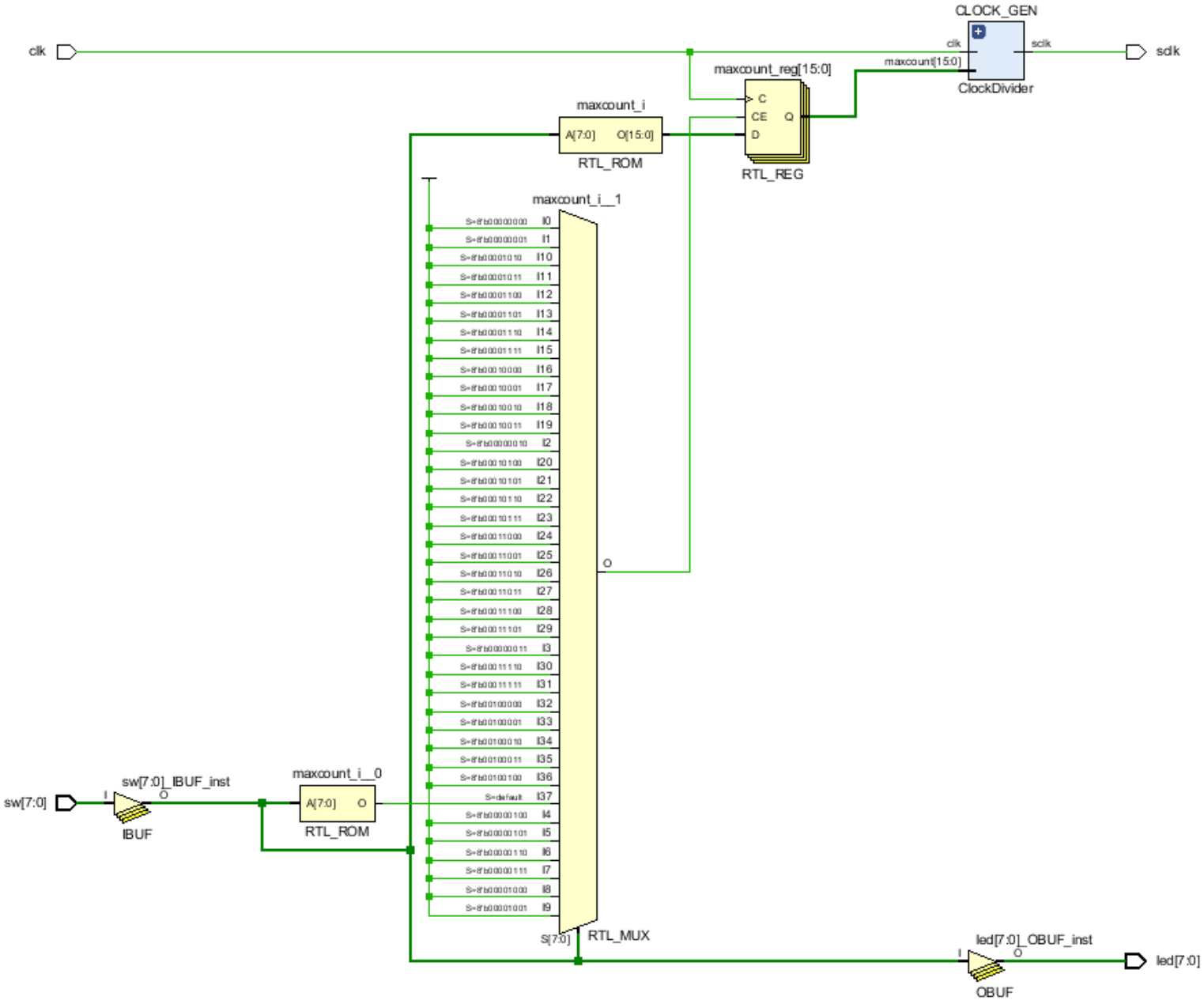


Behavior

The Speaker Driver Interface is a System Verilog module which utilizes an input clock signal and an 8-bit switch input, to output an 8-bit LED buffer and a clock signal of variable frequency (50% duty cycle). Currently, the module is programed to provide 35 musical notes ranging from Octaves 6 -8.

Structural

The structural design of the Speaker Driver Interface is found below. We would like to make note of the large MUX seen in the design. The large size of the MUX accounts for the 35 notes currently programmed in the module.



Specifications

Operating Speed: 100Mhz	**8 convenient LEDS** to indicate enabled switches
Input Clock: 100Mhz	**Volume Contro**l via a 10kohm potentiometer
Generates Output Frequencies with **less than .02 % error!**	**Input Voltage:** 3.3 V

The **Speaker Driver Interface** is a System Verilog module which utilizes an input clock signal and an 8-bit switch input, to output an 8-bit LED buffer and a clock signal of variable frequency (50% duty cycle). Currently, the module is programed to provide 35 musical notes ranging from Octaves 6 -8; however, our input width can support up to 255 different inputs so many more notes can be programmed by the user. The module is equipped with a separate clock divider module that produces a varying clock signal (**sclock**) with a frequency dependent on the desired note. The clock divider receives a 15-bit input (**maxcount**) which is computed according to the following formula:

$$maxcount = \frac{100Mhz}{2 * frequency}$$

The resulting clock signal is then output to an external speaker/amplifier circuit to produce audible sound. For convenience, the module illuminates the 8 LEDS corresponding to toggled input switches on the Basys 3 FPGA. Furthermore, the circuit allows for volume control via a 10kohm potentiometer. The circuit diagram is found below:

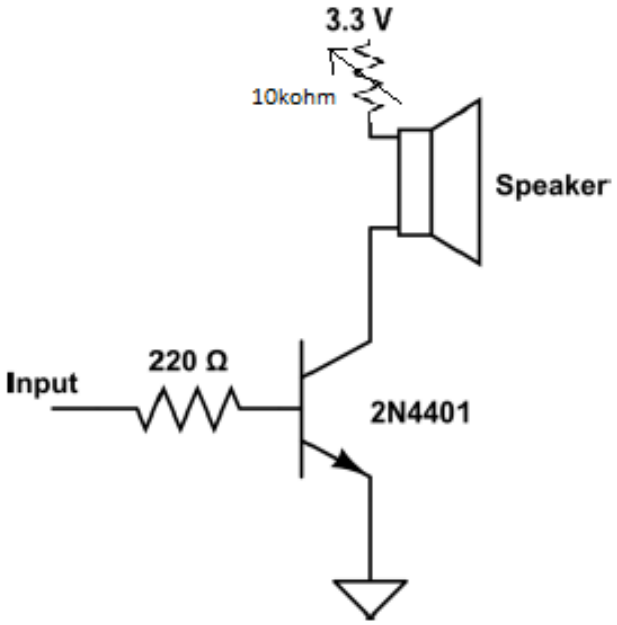
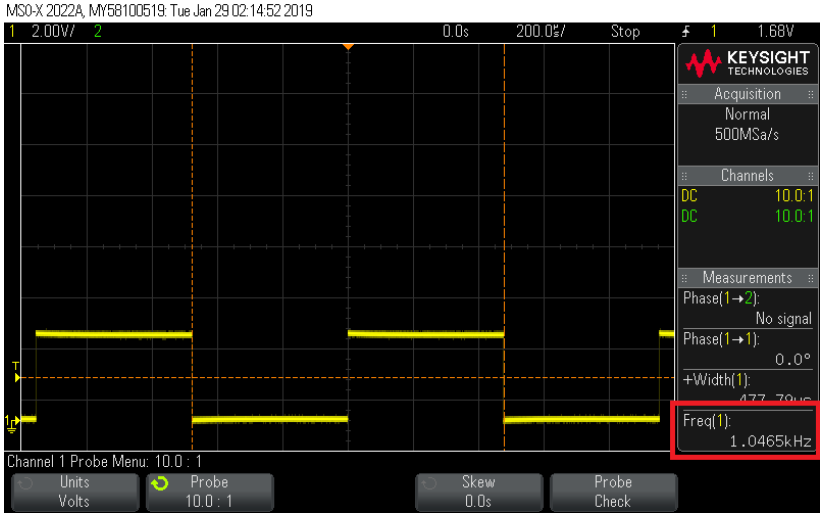


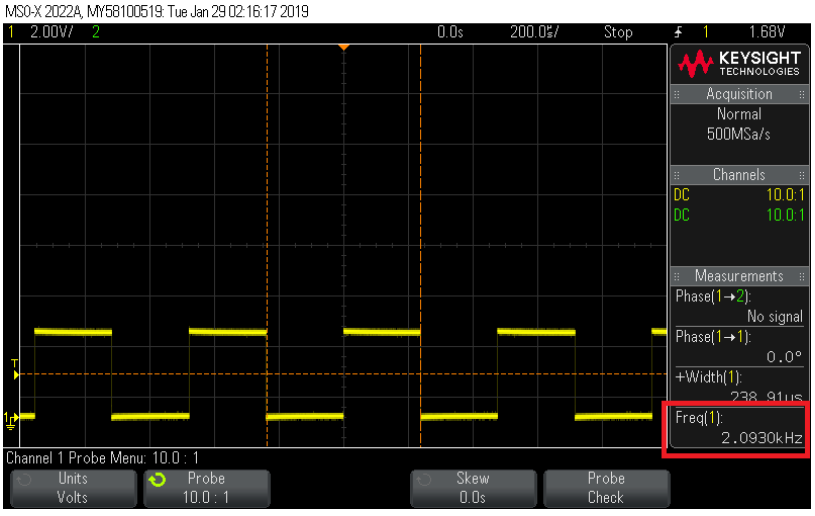
Figure 1: Speaker Driver Circuit

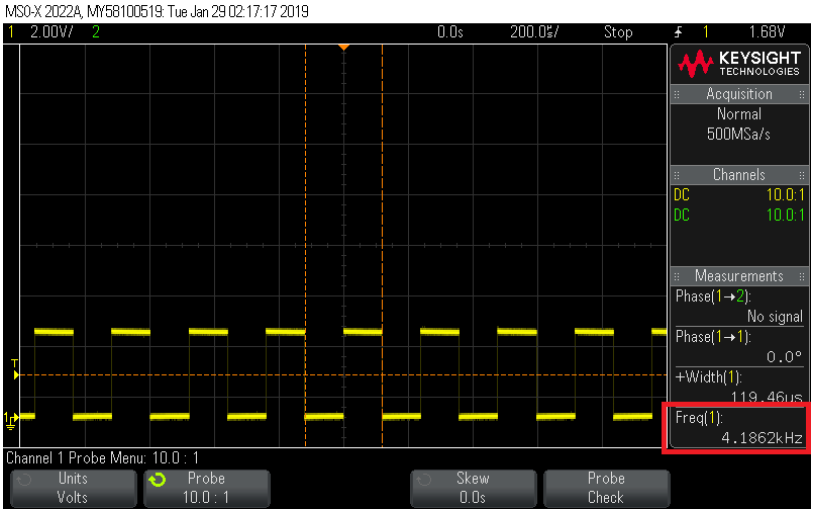
Verification

Input Value	Note	Octave	Frequency (Hz)	maxcount	Verification (Hz)	% error
0	none	none	0	0	0.0	0.00000%
1	C	6	1046.502	47778	1046.6	0.00936%
2	C #, Db	6	1108.731	45097	1108.7	0.00280%
3	D	6	1174.659	42566	1174.7	0.00349%
4	D#, Eb	6	1244.508	40177	1244.5	0.00064%
5	E	6	1318.51	37922	1318.6	0.00683%
6	F	6	1396.913	35793	1397.0	0.00623%
7	F#, Gb	6	1479.978	33784	1480.0	0.00149%
8	G	6	1567.982	31888	1568.0	0.00115%
9	G#, Ab	6	1661.219	30098	1661.4	0.01090%
10	A	6	1760	28409	1760.1	0.00568%
11	A#, Bb	6	1864.655	26815	1864.6	0.00295%
12	B	6	1975.533	25310	1975.5	0.00167%
13	C	7	2093.004	23889	2093.1	0.00459%
14	C #, Db	7	2217.462	22548	2217.5	0.00171%
15	D	7	2349.318	21283	2349.4	0.00349%
16	D#, Eb	7	2489.016	20088	2489.0	0.00064%
17	E	7	2637.02	18961	2637.0	0.00076%
18	F	7	2793.826	17897	2793.9	0.00265%
19	F#, Gb	7	2959.956	16892	2960.0	0.00149%
20	G	7	3135.964	15944	3136.0	0.00115%
21	G#, Ab	7	3322.438	15049	3322.4	0.00114%
22	A	7	3520	14205	3520.1	0.00284%
23	A#, Bb	7	3729.31	13407	3729.5	0.00509%
24	B	7	3951.066	12655	3951.1	0.00086%
25	C	8	4186.008	11945	4186.2	0.00459%
26	C #, Db	8	4434.924	11274	4435.0	0.00171%
27	D	8	4698.636	10641	4698.7	0.00136%
28	D#, Eb	8	4978.032	10044	4978.1	0.00137%
29	E	8	5274.04	9480	5274.3	0.00493%
30	F	8	5587.652	8948	5587.8	0.00265%
31	F#, Gb	8	5919.912	8446	5920.0	0.00149%
32	G	8	6271.928	7972	6272.0	0.00115%
33	G#, Ab	8	6644.876	7525	6645.4	0.00789%
34	A	8	7040	7102	7040.3	0.00426%
35	A#, Bb	8	7458.62	6704	7459.6	0.01314%
36	B	8	7902.132	6327	7902.9	0.00972%

Example Verification, C in 3 octaves. Note the frequency in the Red Box







System Verilog Source Code

SPEAKER DRIVER INTERFACE (4 pgs.)

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//
// Engineer: Luis Gomez
//
// Create Date: 01/26/2019 10:50:59 PM
// Module Name: speaker_driver_interface
// Description:
//   Interface for a Speaker Driver. Provides 8-bit input via
//   switches on the Basys3 board
//   Outputs a 50% duty cycle square wave of variable frequency,
//   according do desired Note & Octave
//   Dependencies: clock_div.sv
//////////////////////////////////////////////////////////////////////////////////

module speaker_driver_interface(
    input clk,
    input [7:0] sw,
    output [7:0] led,
    output sclk // square_wave
    );
    assign sw = led; // enables leds for switches
    logic [15:0] maxcount = 0;

    ClockDivider CLOCK_GEN (.*);

    parameter
    C = 47778, C_sharp = 45097,
    D = 42566, D_sharp = 40177,
    E = 37922,
    F = 35793, F_sharp = 33784,
    G = 31888, G_sharp = 30098,
    A = 28409, A_sharp = 26815,
    B = 25310;
```

```
always_ff @(posedge clk)
    begin
        case (sw)
            0:  begin
                maxcount <= 0;
                end
/**********6th OCTAVE*****************/
            1:  begin   // C
                maxcount <= C;
                end
            2:  begin   // C#, D flat
                maxcount <= C_sharp;
                end
            3:  begin   // D
                maxcount <= D;
                end
            4:  begin   // D#, E flat
                maxcount <= D_sharp;
                end
            5:  begin   // E
                maxcount <= E;
                end
            6:  begin   // F
                maxcount <= F;
                end
            7:  begin // F#, G flat
                maxcount <= F_sharp;
                end
            8:  begin   // G
                maxcount <= G;
                end
            9:  begin   // G#,A flat
                maxcount <= G_sharp;
                end
           10:  begin   // A
                maxcount <= A;
                end
           11:  begin   // A#, B flat
                maxcount <= A_sharp;
                end
           12:  begin   // B
                maxcount <= B;
                end
```

```
/**********7th OCTAVE*****************/
            13:  begin   // C
                 maxcount <= C/2;
                 end
            14:  begin   // C#, D flat
                 maxcount <= C_sharp/2;
                 end
            15:  begin   // D
                 maxcount <= D/2;
                 end
            16:  begin   // D#, E flat
                 maxcount <= D_sharp/2;
                 end
            17:  begin   // E
                 maxcount <= E/2;
                 end
            18:  begin   // F
                 maxcount <= F/2;
                 end
            19:  begin // F#, G flat
                 maxcount <= F_sharp/2;
                 end
            20:  begin   // G
                 maxcount <= G/2;
                 end
            21:  begin   // G#,A flat
                 maxcount <= G_sharp/2;
                 end
            22:  begin   // A
                 maxcount <= A/2;
                 end
            23:  begin   // A#, B flat
                 maxcount <= A_sharp/2;
                 end
            24:  begin   // B
                 maxcount <= B/2;
                 end
```

```
/**********8th OCTAVE*****************/
            25:  begin   // C
                 maxcount <= C/4;
                 end
            26:  begin   // C#, D flat
                 maxcount <= C_sharp/4;
                 end
            27:  begin   // D
                 maxcount <= D/4;
                 end
            28:  begin   // D#, E flat
                 maxcount <= D_sharp/4;
                 end
            29:  begin   // E
                 maxcount <= E/4;
                 end
            30:  begin   // F
                 maxcount <= F/4;
                 end
            31:  begin // F#, G flat
                 maxcount <= F_sharp/4;
                 end
            32:  begin   // G
                 maxcount <= G/4;
                 end
            33:  begin   // G#,A flat
                 maxcount <= G_sharp/4;
                 end
            34:  begin   // A
                 maxcount <= A/4;
                 end
            35:  begin   // A#, B flat
                 maxcount <= A_sharp/4;
                 end
            36:  begin   // B
                 maxcount <= B/4;
                 end
          default:  begin
                 maxcount = 0;
                 end
        endcase
    end
endmodule
```

CLOCK DIVIDER

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Engineer: Bridget Benson
//
// Create Date: 10/01/2018 10:22:13 AM
// Description: Generic Clock Divider.  Divides the input clock by MAXCOUNT*2
//
//////////////////////////////////////////////////////////////////////////////////

module ClockDivider (
    input clk,
    input [15:0] maxcount,
    output logic sclk = 0
    );

    logic [15:0] count = 0;

    always_ff @ (posedge clk)
    begin
        if (maxcount == 0) // LOW CLOCK, I added this to account for 0 input and default cases
            sclk = 0;
        else
            count = count + 1;
            if (count == maxcount)
            begin
                count = 0;
                sclk = ~sclk;
            end
    end

endmodule
```

RAT Assembly Example Use Code

```
;-----------------------------------------------------------------
; Peripheral_one
; by Luis Gomez, Jared Roscha
; date : 1/28/19
;
; Drunken Sailor: A,    A,    A,   A,    A,    A,   A,   D,   F,  A
;          Beats: .25, .125, .125, .25, .125, .125, .25, .25, .25. 25
;    "What shall we do with the Drunken Sailor..."
;-----------------------------------------------------------------
.EQU SPEAKER_OUT = 0x03          ; speaker port on fpga
.CSEG
.ORG 0x01

OUT 0x0A, SPEAKER_OUT            ; A, 1/4 note
;DELAY FUNCTION .25 sec
OUT 0x0A, SPEAKER_OUT            ; A, 1/8 note
;DELAY FUNCTION .125 sec
OUT 0x0A, SPEAKER_OUT            ; A, 1/8 note
;DELAY FUNCTION .125 sec
OUT 0x0A, SPEAKER_OUT            ; A, 1/4 note
;DELAY FUNCTION .25 sec
OUT 0x0A, SPEAKER_OUT            ; A, 1/8 note
;DELAY FUNCTION .125 sec
OUT 0x0A, SPEAKER_OUT            ; A, 1/8 note
;DELAY FUNCTION .125 sec
OUT 0x0A, SPEAKER_OUT            ; A, 1/4 note
;DELAY FUNCTION .25 sec
OUT 0x03, SPEAKER_OUT            ; D, 1/4 note
;DELAY FUNCTION .25 sec
OUT 0x06, SPEAKER_OUT            ; F, 1/4 note
;DELAY FUNCTION .25 sec
OUT 0x0A, SPEAKER_OUT            ; A, 1/4 note
;DELAY FUNCTION .25 sec
```