EECS2070 02
Digilent Basys3 FPGA Board
Part 6

Ref: Digilent Basys3™ FPGA Board Reference Manual

黃稚存

國立清華大學
資訊工程學系

Lecture 14

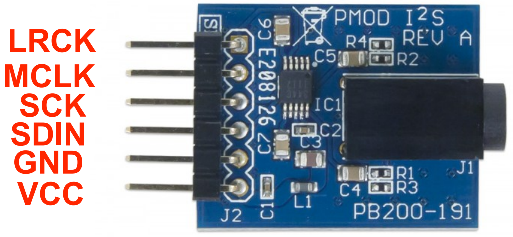
Pmod I2S

Stereo Audio Amplifier

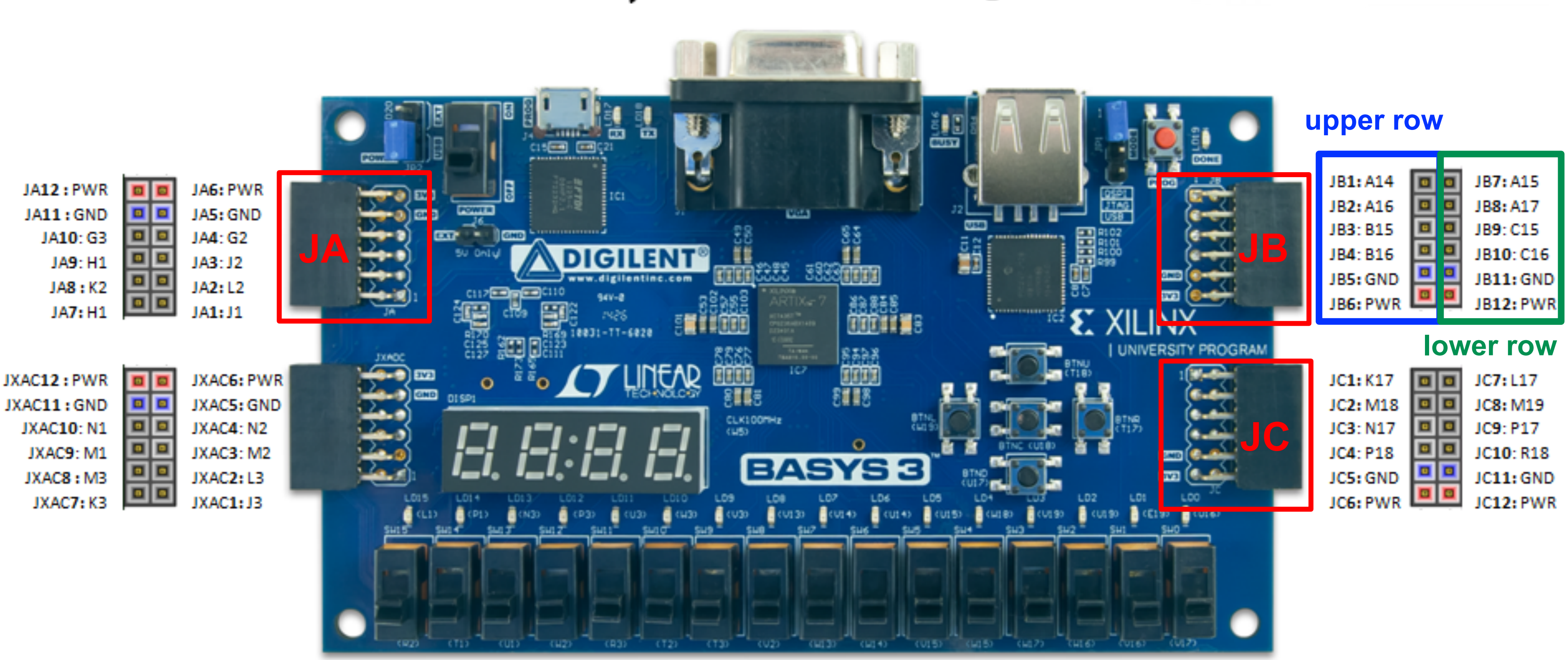
Pmod Connectors (1/2)

- Basys3 provides 3 Pmod connectors for user

Pmod I2S



Basys3: Pmod Pin-Out Diagram



JA12 : PWR	JA6: PWR
JA11 : GND	JA5: GND
JA10: G3	JA4: G2
JA9: H1	JA3: J2
JA8 : K2	JA2: L2
JA7: H1	JA1: J1

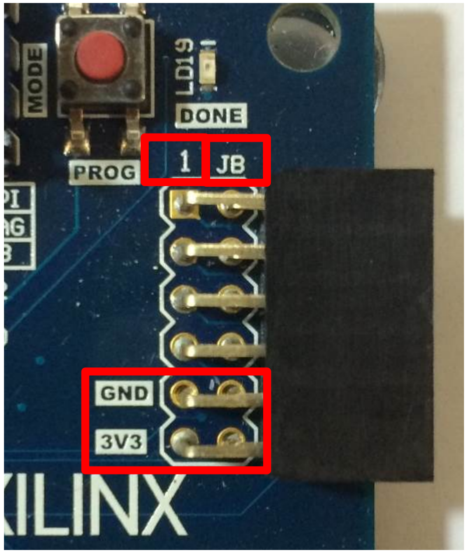
JXAC12 : PWR	JXAC6: PWR
JXAC11 : GND	JXAC5: GND
JXAC10: N1	JXAC4: N2
JXAC9: M1	JXAC3: M2
JXAC8 : M3	JXAC2: L3
JXAC7: K3	JXAC1: J3

JB1: A14	JB7: A15
JB2: A16	JB8: A17
JB3: B15	JB9: C15
JB4: B16	JB10: C16
JB5: GND	JB11: GND
JB6: PWR	JB12: PWR

JC1: K17	JC7: L17
JC2: M18	JC8: M19
JC3: N17	JC9: P17
JC4: P18	JC10: R18
JC5: GND	JC11: GND
JC6: PWR	JC12: PWR

Pmod Connectors (2/2)

- Each 12-pin Pmod connector provides
  - **Two** 3.3V VCC signals (pins 6 and 12)
  - **Two** Ground signals (pins 5 and 11)
  - **Eight** logic signals



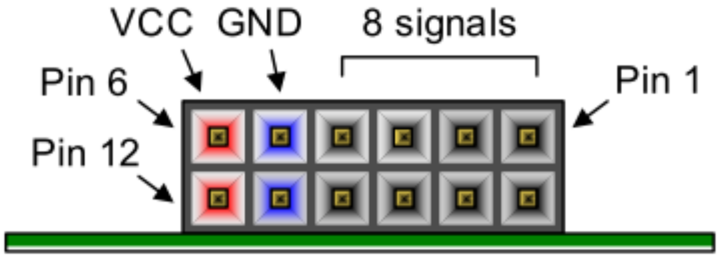
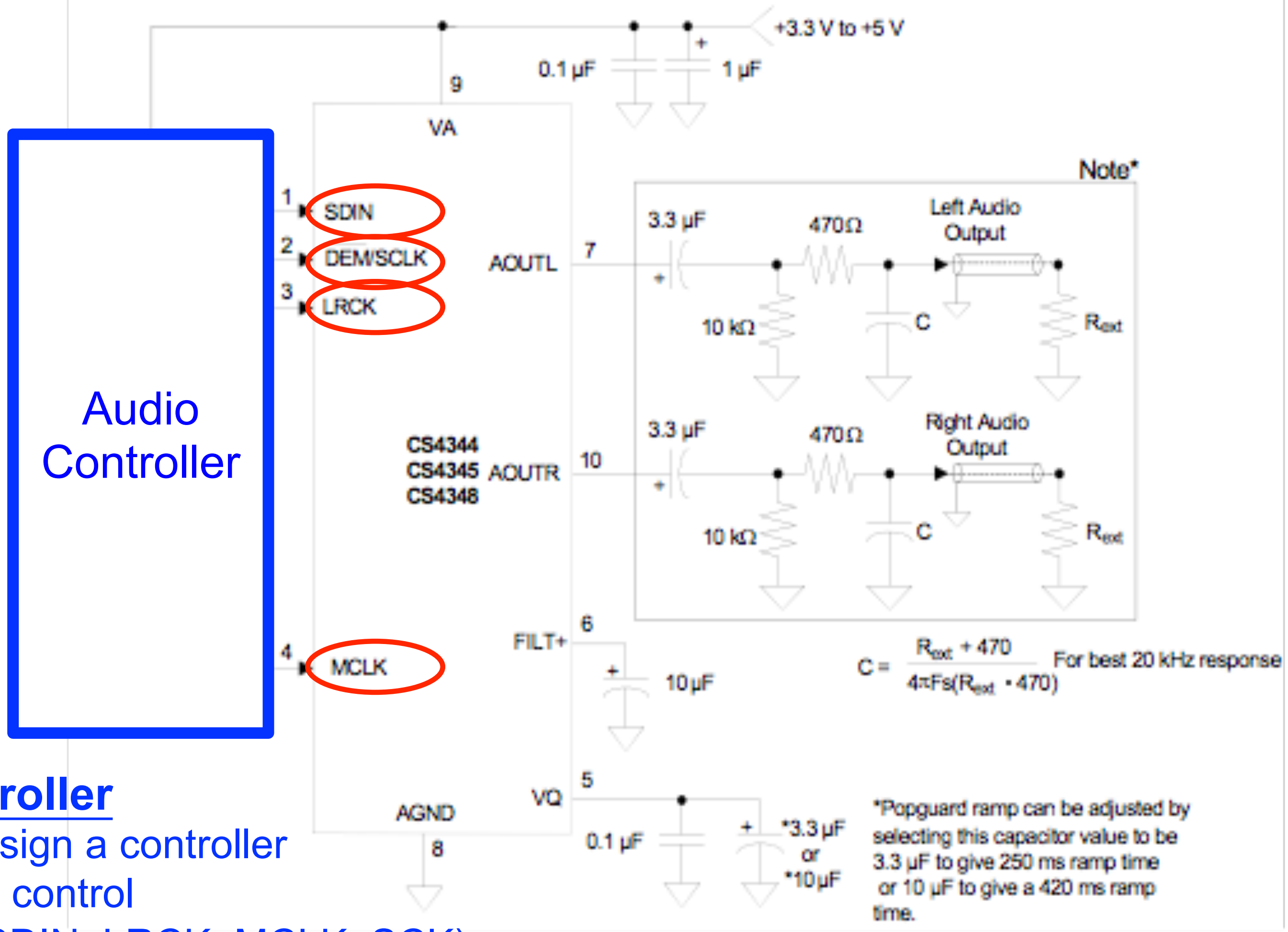


Figure 20. Pmod connectors; front view as loaded on PCB.

Speaker

- Control the DAC (digital to analog converter) CS4344

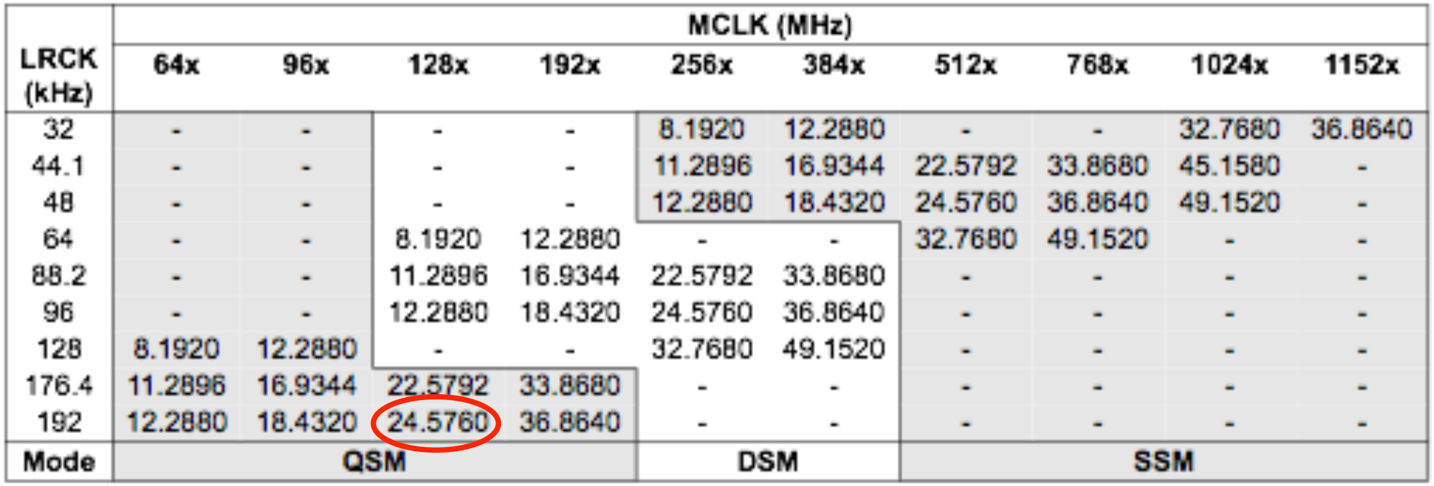


Audio Controller

We should design a controller for audio data control (to generate SDIN, LRCK, MCLK, SCK)

Preset Parameters

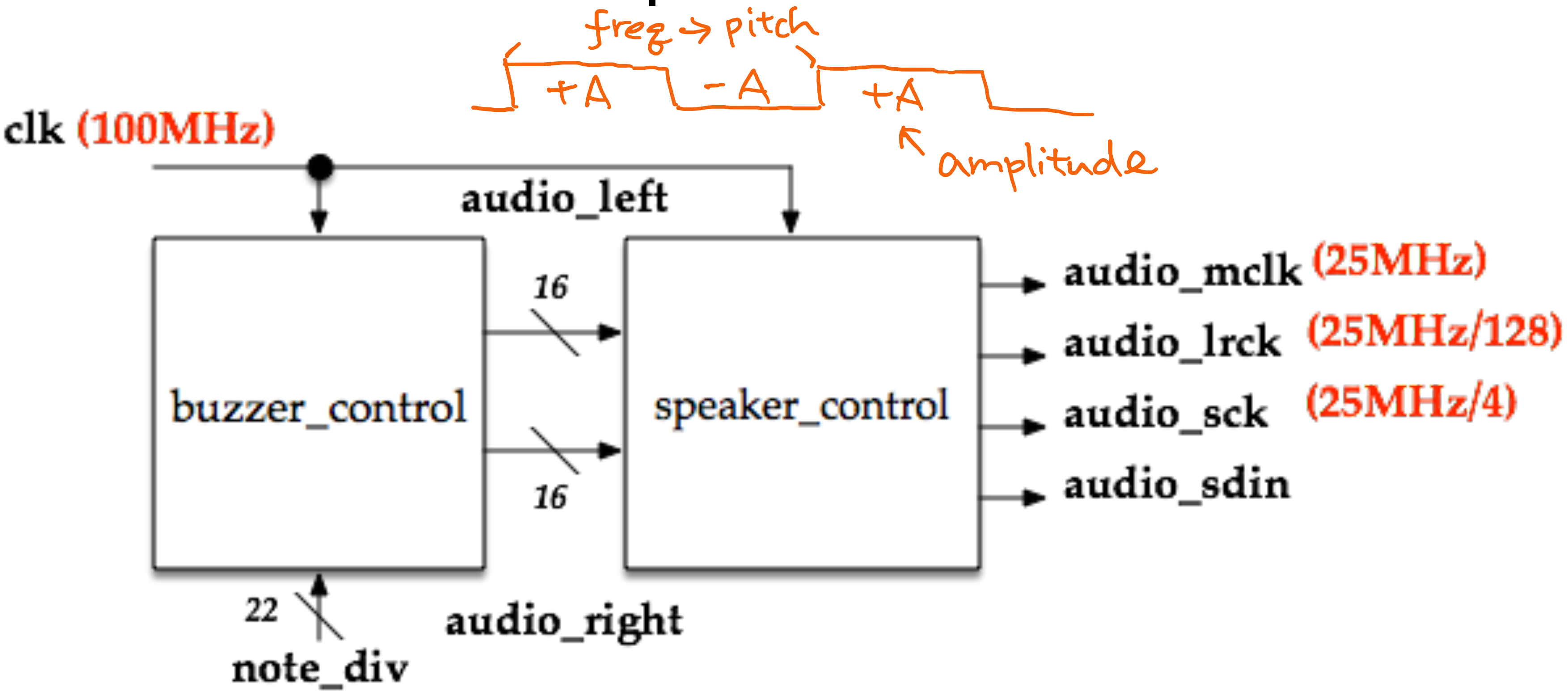
- Control the DAC (digital to analog converter) CS4344
  - LRCK (Left-Right Clock, or Word Select (WS) Clock, or Sample Rate (Fs) Clock) controls the sequence (left or right) of the serial stereo output
    - 25MHz/128 (~192kHz)
  - MCLK (Master Clock) synchronizes the audio data transmission
    - 25MHz (~24.5760MHz)
  - MCLK/LRCK must be an integer ratio
    - 128
  - Serial Clock (SCK) controls the shifting of data into the input data buffers (32*Fs)
    - 25MHz/128*32 = 25MHz/4



LRCK (kHz)	MCLK (MHz)									
	64x	96x	128x	192x	256x	384x	512x	768x	1024x	1152x
32	-	-	-	-	8.1920	12.2880	-	-	32.7680	36.8640
44.1	-	-	-	-	11.2896	16.9344	22.5792	33.8680	45.1580	-
48	-	-	-	-	12.2880	18.4320	24.5760	36.8640	49.1520	-
64	-	-	8.1920	12.2880	-	-	32.7680	49.1520	-	-
88.2	-	-	11.2896	16.9344	22.5792	33.8680	-	-	-	-
96	-	-	12.2880	18.4320	24.5760	36.8640	-	-	-	-
128	8.1920	12.2880	-	-	32.7680	49.1520	-	-	-	-
176.4	11.2896	16.9344	22.5792	33.8680	-	-	-	-	-	-
192	12.2880	18.4320	24.5760	36.8640	-	-	-	-	-	-
Mode	QSM				DSM		SSM			

Audio Controller

- Buzzer control + speaker control



Buzzer Control

- The buzzer frequency is obtained by dividing crystal frequency 100MHz by *N*
  - E.g., cnt_max = 100_000_000 / *N*
- The buzzer clock (b_clk) is periodically inverted for every *N*/2 clock cycles
  - ***To determine the sound frequency***
- Example for music notes
  - Mid Do: 261 Hz
  - Mid Re: 293 Hz
  - Mid Mi: 330 Hz

Ex: Do → 261 Hz

$100{,}000{,}000/N = 261$

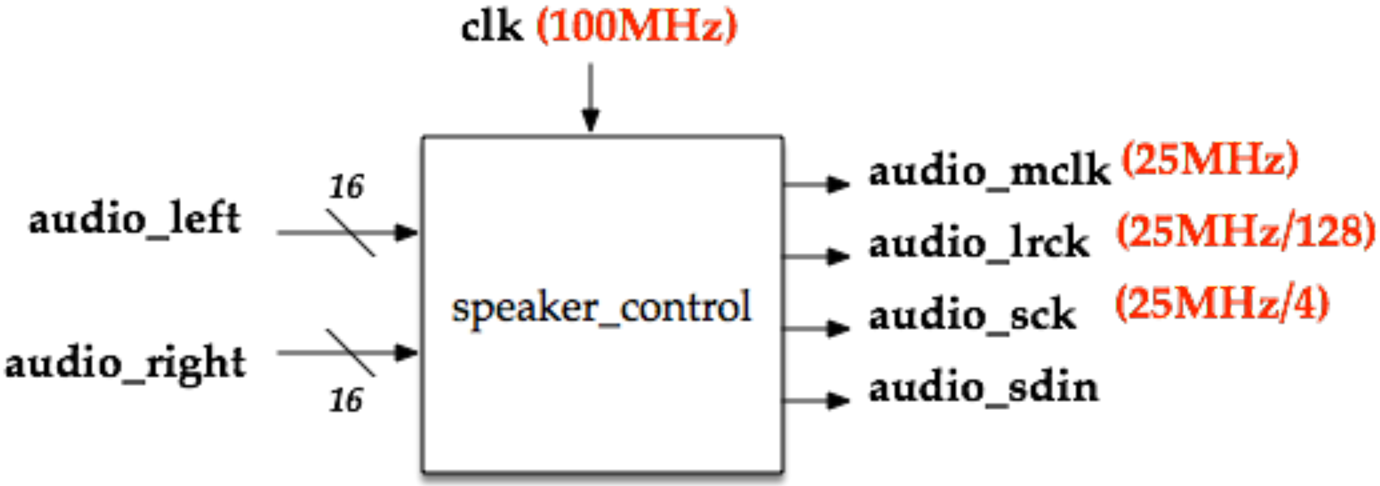
$N = 383{,}142$

$N/2 = 191{,}571$

Speaker Control

- Inputs (stereo audio ***parallel input***)
  - Two 16-bit channels: audio_left [15:0], audio_right[15:0]
  - Value: 16‘h8000 (min) < *V* < 16‘h7FFF (max)
    - Volume of output sound in 16-bit two’s complement
    - Keep $|V_{max}| \approx |V_{min}|$
- Outputs (stereo audio ***serial output***)
  - audio_mclk = 25MHz (divided by 4 from external 100MHz clock source)
  - audio_lrck = 25MHz/128 (clock rate of parallel audio input)
  - audio_sck = 25MHz/4 (clock rate of serial audio output)
  - audio_sdin (1 bit serial audio data output)

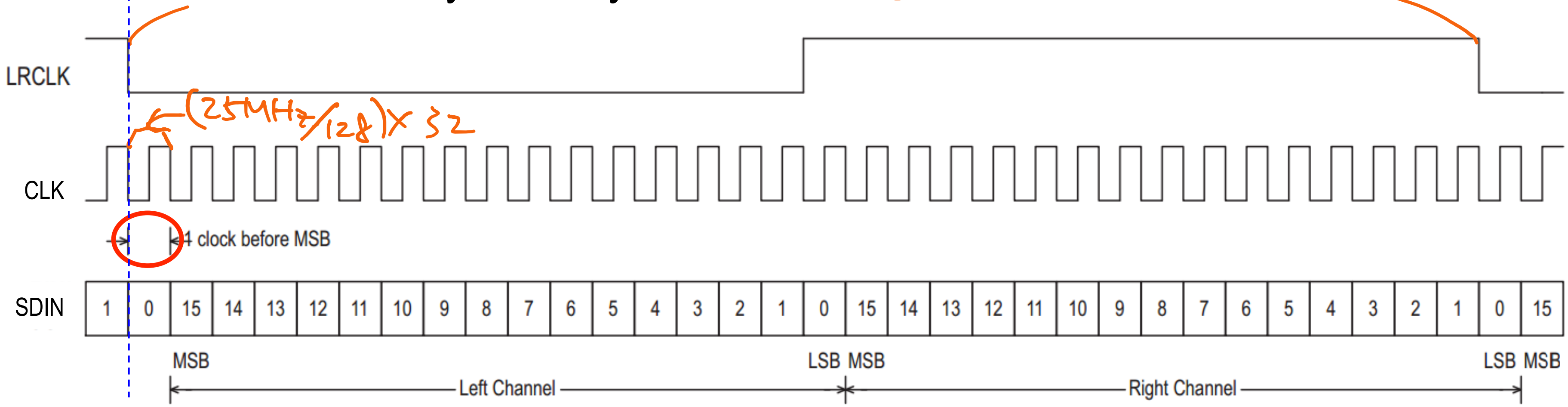
Data rate is the same: 6.25Mbps



Speaker Control

- Frequency dividers
  - audio_mclk
  - audio_lrck
  - audio_sck
- Parallel to serial module
  - To re-formulate the audio sequence
    - Left first, then right
    - MSB first
    - one SCK cycle delayed

25MHz/128


(25MHz/128)×32

Buzzer Control

```
module buzzer_control(
  clk, // clock from crystal
  rst, // active high reset
  note_div, // div for note generation
  audio_left, // left sound audio
  audio_right // right sound audio
);

// I/O declaration
input clk; // clock from crystal
input rst; // active high reset
input [21:0] note_div; // div for note generation
output [15:0] audio_left; // left sound audio
output [15:0] audio_right; // right sound audio

// Declare internal signals
reg [21:0] clk_cnt_next, clk_cnt;
reg b_clk, b_clk_next;
```

```
// Note frequency generation
always @(posedge clk or posedge rst)
  if (rst == 1'b1) begin
    clk_cnt <= 22'd0;
    b_clk <= 1'b0;
  end else begin
    clk_cnt <= clk_cnt_next;
    b_clk <= b_clk_next;
  end
```

191,571

```
always @*
  if (clk_cnt == note_div) begin
    clk_cnt_next = 22'd0;
    b_clk_next = ~b_clk;
  end else begin
    clk_cnt_next = clk_cnt + 1'b1;
    b_clk_next = b_clk;
  end
```

Ex: $\begin{cases} -2^{13} \rightarrow \text{hE000} \\ +2^{13} \rightarrow \text{h2000} \end{cases}$

-2^{14} 2^{14}

```
// Assign the amplitude of the note
assign audio_left =
    (b_clk == 1'b0) ? 16’hC000 : 16’h4000;
assign audio_right =
    (b_clk == 1'b0) ? 16’hC000 : 16’h4000;
endmodule
```

Speaker Control (1/2)

```
module speaker_control(
  clk,  // clock from the crystal
  rst,  // active high reset
  audio_in_left,  // left audio data input
  audio_in_right, // right channel audio input
  audio_mclk, // master clock
  audio_lrck, // left-right clock, Word Select
              // clock, or sample rate clock
  audio_sck,  // serial clock
  audio_sdin  // serial audio data input
);

// I/O declaration
input clk;  // clock from the crystal
input rst;  // active high reset
input [15:0] audio_in_left; // left audio input
input [15:0] audio_in_right; // right audio input
output audio_mclk; // master clock
output audio_lrck; // left-right clock
output audio_sck; // serial clock
output audio_sdin; // serial audio data input
reg audio_sdin;
```

```
// Declare internal signal nodes
wire [8:0] clk_cnt_next;
reg [8:0] clk_cnt;
reg [15:0] audio_left, audio_right;

// Counter for the clock divider
assign clk_cnt_next = clk_cnt + 1'b1;

always @(posedge clk or posedge rst)
  if (rst == 1'b1)
    clk_cnt <= 9'd0;            100MHz
  else
    clk_cnt <= clk_cnt_next;

// Assign divided clock output
assign audio_mclk = clk_cnt[1];   clk/4 = 25MHz
assign audio_lrck = clk_cnt[8];   clk/512 = 25MHz/128

// use internal serial clock mode
assign audio_sck = 1'b1;
```

Speaker Control (2/2)

```
// audio input data buffer
always @(posedge clk_cnt[8] or posedge rst)
  if (rst == 1’b1) begin
    audio_left <= 16'd0;
    audio_right <= 16'd0;
  end else begin
    audio_left <= audio_in_left;
    audio_right <= audio_in_right;
  end
```

$lrck = \frac{25MHz}{128}$

$sck = \frac{25MHz}{128} \times 32$

```
always @*
  case (clk_cnt[8:4])
    5'b00000: audio_sdin = audio_right[0];
    5'b00001: audio_sdin = audio_left[15];
    5'b00010: audio_sdin = audio_left[14];
    5'b00011: audio_sdin = audio_left[13];
    5'b00100: audio_sdin = audio_left[12];
    5'b00101: audio_sdin = audio_left[11];
    5'b00110: audio_sdin = audio_left[10];
    5'b00111: audio_sdin = audio_left[9];
    5'b01000: audio_sdin = audio_left[8];
    5'b01001: audio_sdin = audio_left[7];
    5'b01010: audio_sdin = audio_left[6];
    5'b01011: audio_sdin = audio_left[5];
    5'b01100: audio_sdin = audio_left[4];
    5'b01101: audio_sdin = audio_left[3];
    5'b01110: audio_sdin = audio_left[2];
    5'b01111: audio_sdin = audio_left[1];
    5'b10000: audio_sdin = audio_left[0];
    5'b10001: audio_sdin = audio_right[15];
    5'b10010: audio_sdin = audio_right[14];
    5'b10011: audio_sdin = audio_right[13];
    5'b10100: audio_sdin = audio_right[12];
    5'b10101: audio_sdin = audio_right[11];
    5'b10110: audio_sdin = audio_right[10];
    5'b10111: audio_sdin = audio_right[9];
    5'b11000: audio_sdin = audio_right[8];
    5'b11001: audio_sdin = audio_right[7];
    5'b11010: audio_sdin = audio_right[6];
    5'b11011: audio_sdin = audio_right[5];
    5'b11100: audio_sdin = audio_right[4];
    5'b11101: audio_sdin = audio_right[3];
    5'b11110: audio_sdin = audio_right[2];
    5'b11111: audio_sdin = audio_right[1];
    default: audio_sdin = 1'b0;
  endcase

endmodule
```

Top Model of Audio Controller: speaker.v

```
module speaker(
  clk, // clock from crystal
  rst, // active high reset
  audio_mclk, // master clock
  audio_lrck, // left-right clock
  audio_sck, // serial clock
  audio_sdin // serial audio data input
);

// I/O declaration
input clk;  // clock from the crystal
input rst;  // active high reset
output audio_mclk; // master clock
output audio_lrck; // left-right clock
output audio_sck; // serial clock
output audio_sdin; // serial audio data input
// Declare internal nodes
wire [15:0] audio_in_left, audio_in_right;

// Note generation
buzzer_control Ung(
  .clk(clk), // clock from crystal
  .rst(rst), // active high reset
  .note_div(22'd191571), // div for note generation
  .audio_left(audio_in_left), // left sound audio
  .audio_right(audio_in_right) // right sound audio
);
```

261Hz

```
// Speaker controllor
speaker_control Usc(
  .clk(clk),  // clock from the crystal
  .rst(rst),  // active high reset
  .audio_in_left(audio_in_left), // left channel
  .audio_in_right(audio_in_right), // right channel
  .audio_mclk(audio_mclk), // master clock
  .audio_lrck(audio_lrck), // left-right clock
  .audio_sck(audio_sck), // serial clock
  .audio_sdin(audio_sdin) // serial audio data
input
);

endmodule
```

Example of Pmod Pin Assignment: speaker.xdc

```
# Clock
set_property PACKAGE_PIN W5 [get_ports {clk}]
set_property IOSTANDARD LVCMOS33 [get_ports {clk}]

# active low reset
set_property PACKAGE_PIN V17 [get_ports {rst}]
set_property IOSTANDARD LVCMOS33 [get_ports {rst}]

# Pmod I2S
set_property PACKAGE_PIN A14 [get_ports {audio_mclk}]
set_property IOSTANDARD LVCMOS33 [get_ports {audio_mclk}]
set_property PACKAGE_PIN A16 [get_ports {audio_lrck}]
set_property IOSTANDARD LVCMOS33 [get_ports {audio_lrck}]
set_property PACKAGE_PIN B15 [get_ports {audio_sck}]
set_property IOSTANDARD LVCMOS33 [get_ports {audio_sck}]
set_property PACKAGE_PIN B16 [get_ports {audio_sdin}]
set_property IOSTANDARD LVCMOS33 [get_ports {audio_sdin}]
```

General Pin Assignments for Pmod Connectors

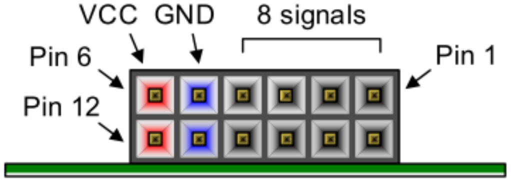
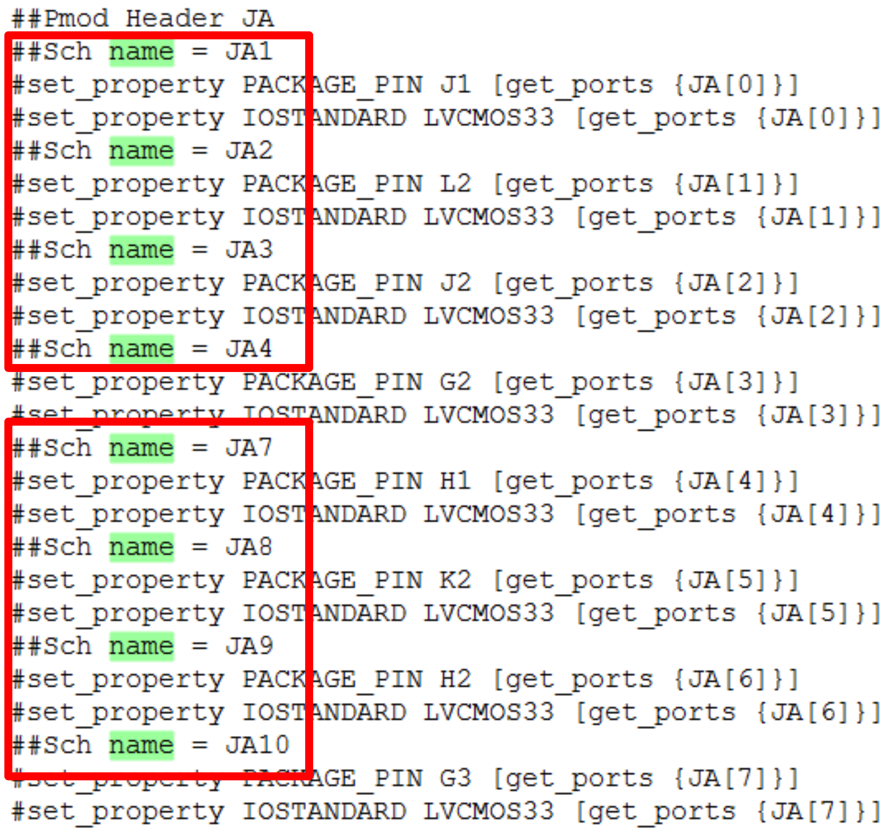
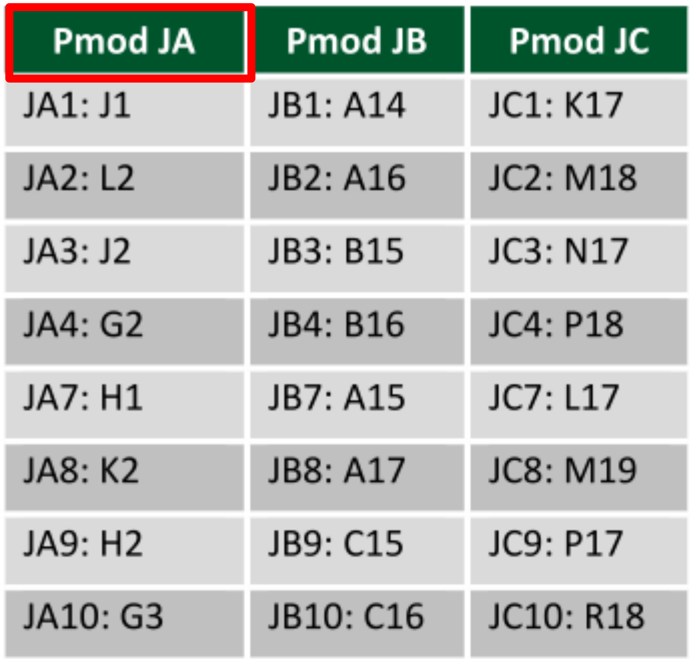


Figure 20. Pmod connectors; front view as loaded on PCB.

- Corresponding pins in a constraint file (.xdc)



```
##Pmod Header JA
##Sch name = JA1
#set_property PACKAGE_PIN J1 [get_ports {JA[0]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JA[0]}]
##Sch name = JA2
#set_property PACKAGE_PIN L2 [get_ports {JA[1]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JA[1]}]
##Sch name = JA3
#set_property PACKAGE_PIN J2 [get_ports {JA[2]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JA[2]}]
##Sch name = JA4
#set_property PACKAGE_PIN G2 [get_ports {JA[3]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JA[3]}]
##Sch name = JA7
#set_property PACKAGE_PIN H1 [get_ports {JA[4]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JA[4]}]
##Sch name = JA8
#set_property PACKAGE_PIN K2 [get_ports {JA[5]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JA[5]}]
##Sch name = JA9
#set_property PACKAGE_PIN H2 [get_ports {JA[6]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JA[6]}]
##Sch name = JA10
#set_property PACKAGE_PIN G3 [get_ports {JA[7]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JA[7]}]
```



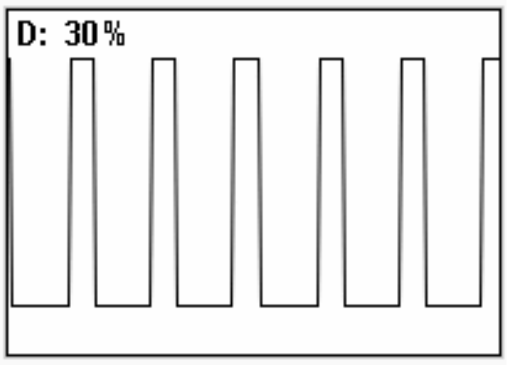
Pmod JA	Pmod JB	Pmod JC
JA1: J1	JB1: A14	JC1: K17
JA2: L2	JB2: A16	JC2: M18
JA3: J2	JB3: B15	JC3: N17
JA4: G2	JB4: B16	JC4: P18
JA7: H1	JB7: A15	JC7: L17
JA8: K2	JB8: A17	JC8: M19
JA9: H2	JB9: C15	JC9: P17
JA10: G3	JB10: C16	JC10: R18

Table 6. Basys3 Pmod pin assignm

音樂課

Basic Concepts of Sound

- **Sound** that is perceptible by humans has frequencies from about 20 Hz to 20,000 Hz
- **Frequency**:
  - The higher frequency, the higher pitch
  - The lower frequency, the lower pitch
- **Duty Cycle**:
  - The evener duty cycle, the better quality
  - ~50%



音高 (pitch)

	Do-H	Re-H	Mi-H	Fa-H	So-H	La-H	Si-H
數字	$\dot{1}$	$\dot{2}$	$\dot{3}$	$\dot{4}$	$\dot{5}$	$\dot{6}$	$\dot{7}$
頻率(Hz)	524	588	660	698	784	880	988

唱名	Do	Re	Mi	Fa	So	La	Si
音高	C	D	E	F	G	A	B
數字	1	2	3	4	5	6	7
頻率(Hz)	262	294	330	349	392	440	494

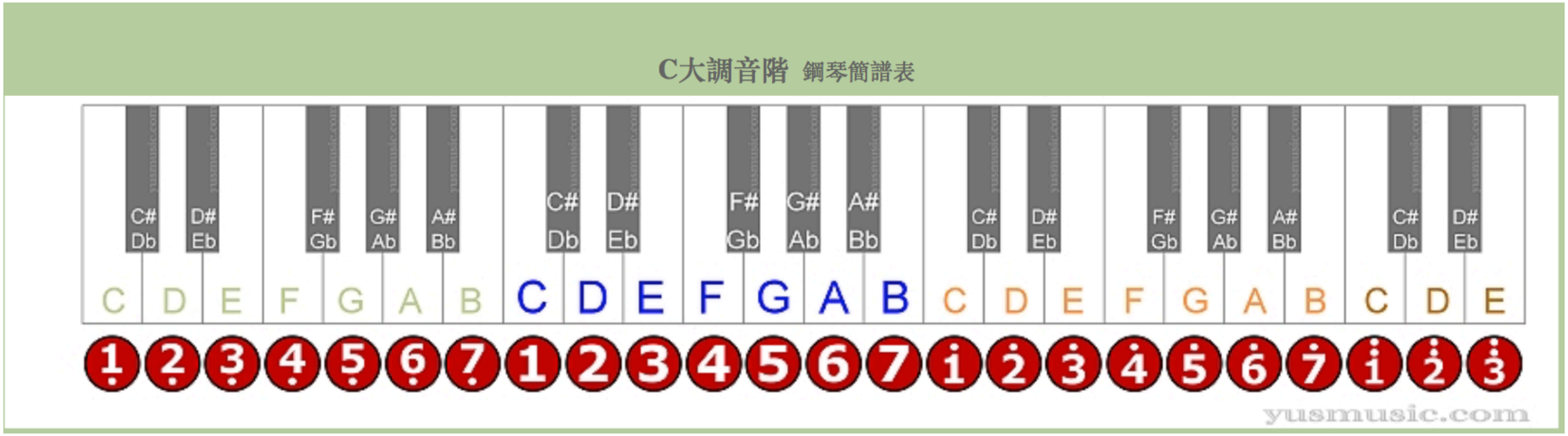
唱名	Do-L	Re-L	Mi-L	Fa-L	So-L	La-L	Si-L
數字	$\underset{\cdot}{1}$	$\underset{\cdot}{2}$	$\underset{\cdot}{3}$	$\underset{\cdot}{4}$	$\underset{\cdot}{5}$	$\underset{\cdot}{6}$	$\underset{\cdot}{7}$
頻率(Hz)	131	147	165	174	196	220	247

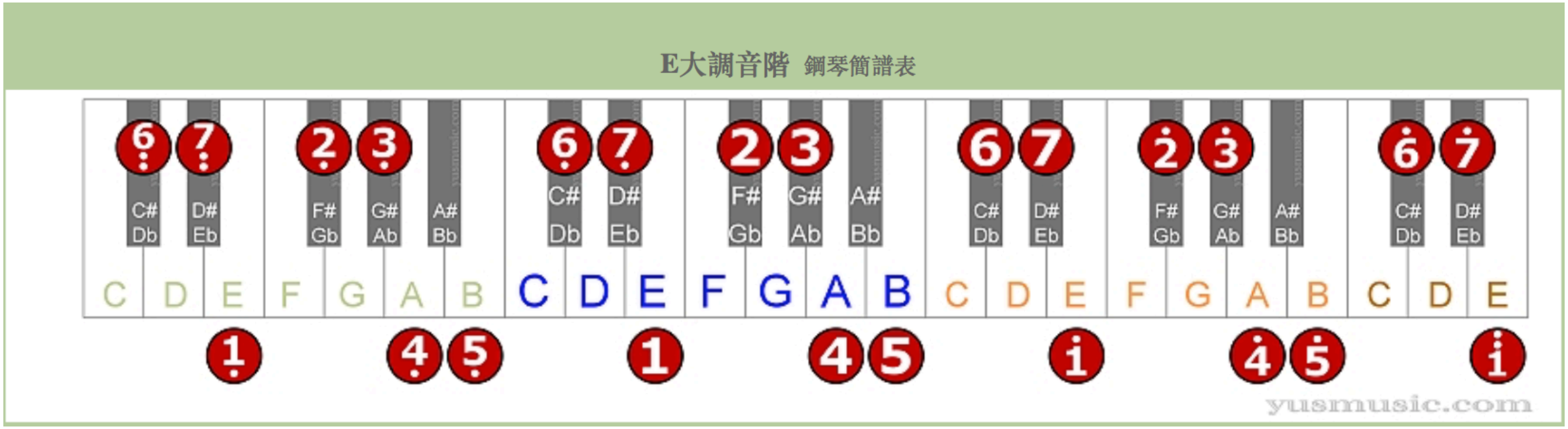
Ref. “音高”, wikipedia

Ref. “Numbered Musical Notation (簡譜)”, wikipedia

鍵盤音名與簡譜對應

(也可以參考超簡單樂理 Lesson 1 嘎老師 Miss Ga https://youtu.be/IiyzCW--nTY)





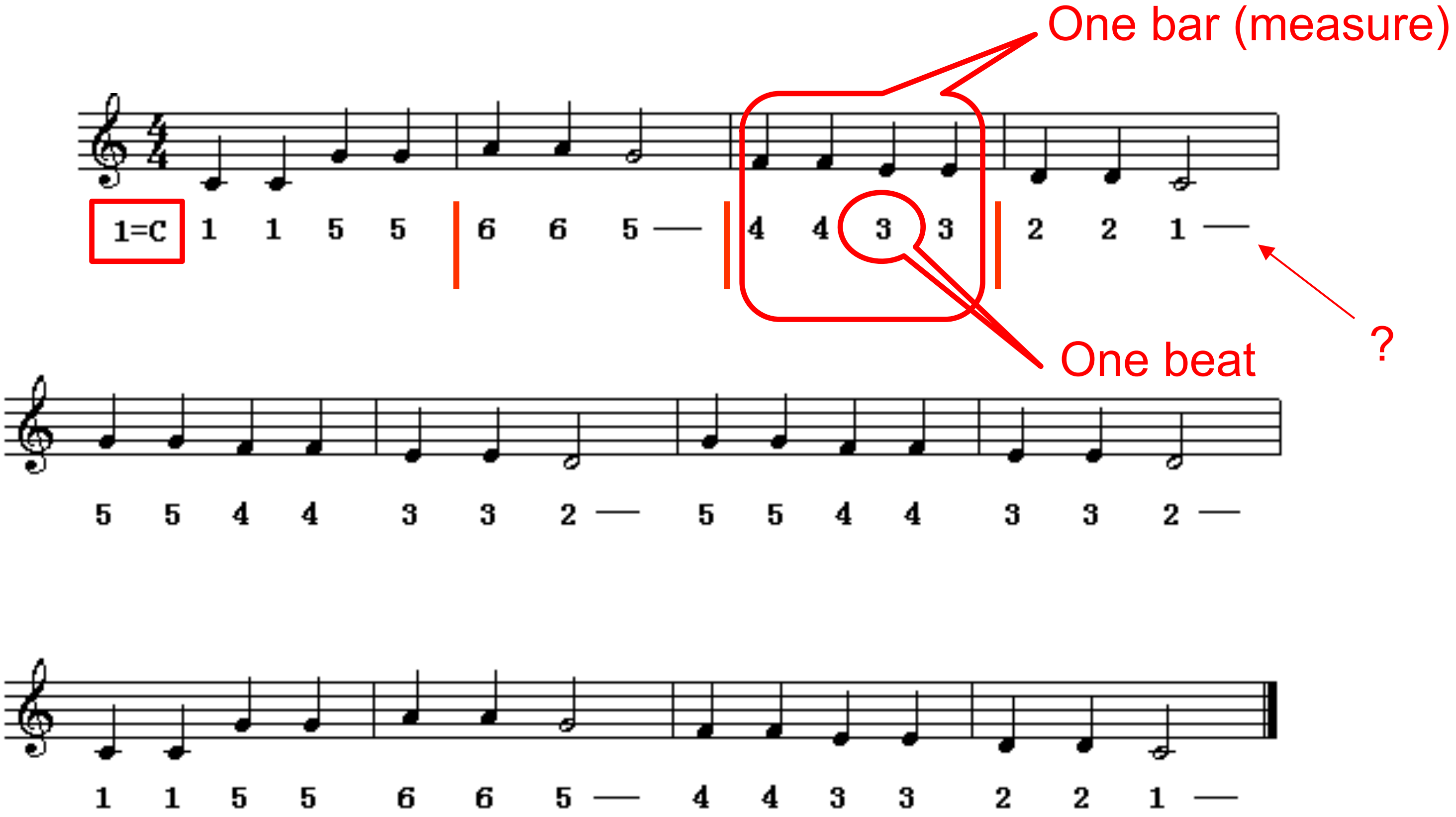
Src: Yus Music: https://goo.gl/4emT4H

頻率，單位為赫茲（括號內為半音距離,"（0）"為中央C）										
C	16.352 (−48)	32.703 (−36)	65.406 (−24)	130.81 (−12)	261.63 (0)	523.25 (+12)	1046.5 (+24)	2093.0 (+36)	4186.0 (+48)	8372.0 (+60)
↑ C♯/D♭ ↓	17.324 (−47)	34.648 (−35)	69.296 (−23)	138.59 (−11)	277.18 (+1)	554.37 (+13)	1108.7 (+25)	2217.5 (+37)	4434.9 (+49)	8869.8 (+61)
D	18.354 (−46)	36.708 (−34)	73.416 (−22)	146.83 (−10)	293.66 (+2)	587.33 (+14)	1174.7 (+26)	2349.3 (+38)	4698.6 (+50)	9397.3 (+62)
D♯/E♭	19.445 (−45)	38.891 (−33)	77.782 (−21)	155.56 (−9)	311.13 (+3)	622.25 (+15)	1244.5 (+27)	2489.0 (+39)	4978.0 (+51)	9956.1 (+63)
E	20.602 (−44)	41.203 (−32)	82.407 (−20)	164.81 (−8)	329.63 (+4)	659.26 (+16)	1318.5 (+28)	2637.0 (+40)	5274.0 (+52)	10548 (+64)
F	21.827 (−43)	43.654 (−31)	87.307 (−19)	174.61 (−7)	349.23 (+5)	698.46 (+17)	1396.9 (+29)	2793.8 (+41)	5587.7 (+53)	11175 (+65)
F♯/G♭	23.125 (−42)	46.249 (−30)	92.499 (−18)	185.00 (−6)	369.99 (+6)	739.99 (+18)	1480.0 (+30)	2960.0 (+42)	5919.9 (+54)	11840 (+66)
G	24.500 (−41)	48.999 (−29)	97.999 (−17)	196.00 (−5)	392.00 (+7)	783.99 (+19)	1568.0 (+31)	3136.0 (+43)	6271.9 (+55)	12544 (+67)
G♯/A♭	25.957 (−40)	51.913 (−28)	103.83 (−16)	207.65 (−4)	415.30 (+8)	830.61 (+20)	1661.2 (+32)	3322.4 (+44)	6644.9 (+56)	13290 (+68)
A	27.500 (−39)	55.000 (−27)	110.00 (−15)	220.00 (−3)	440.00 (+9)	880.00 (+21)	1760.0 (+33)	3520.0 (+45)	7040.0 (+57)	14080 (+69)
A♯/B♭	29.135 (−38)	58.270 (−26)	116.54 (−14)	233.08 (−2)	466.16 (+10)	932.33 (+22)	1864.7 (+34)	3729.3 (+46)	7458.6 (+58)	14917 (+70)
B	30.868 (−37)	61.735 (−25)	123.47 (−13)	246.94 (−1)	493.88 (+11)	987.77 (+23)	1975.5 (+35)	3951.1 (+47)	7902.1 (+59)	15804 (+71)

Ref. “音高”, Wikipedia: https://zh.wikipedia.org/wiki/音高

Numbered Musical Notation 1

小星星



One bar (measure)

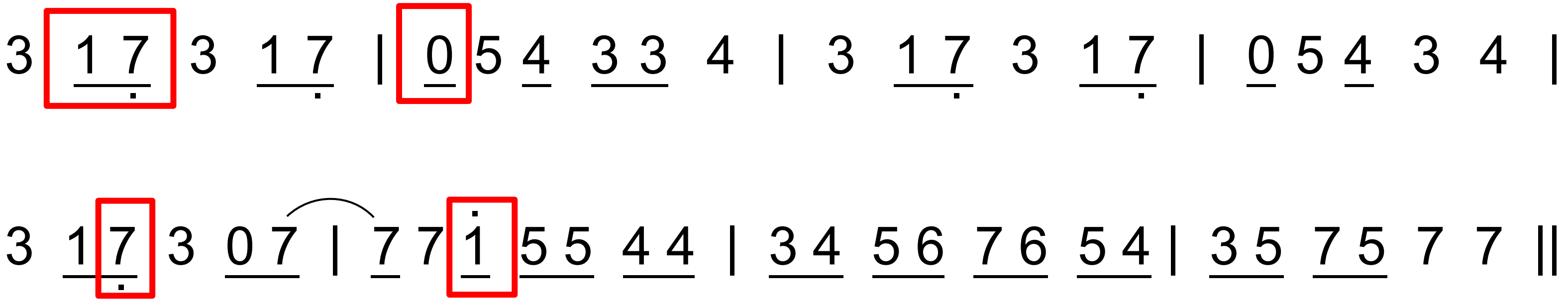
One beat

?

Numbered Musical Notation 2

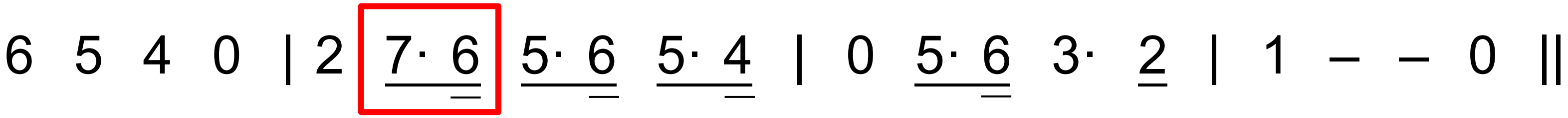
櫻桃小丸子

1=E　4/4



哆啦A夢

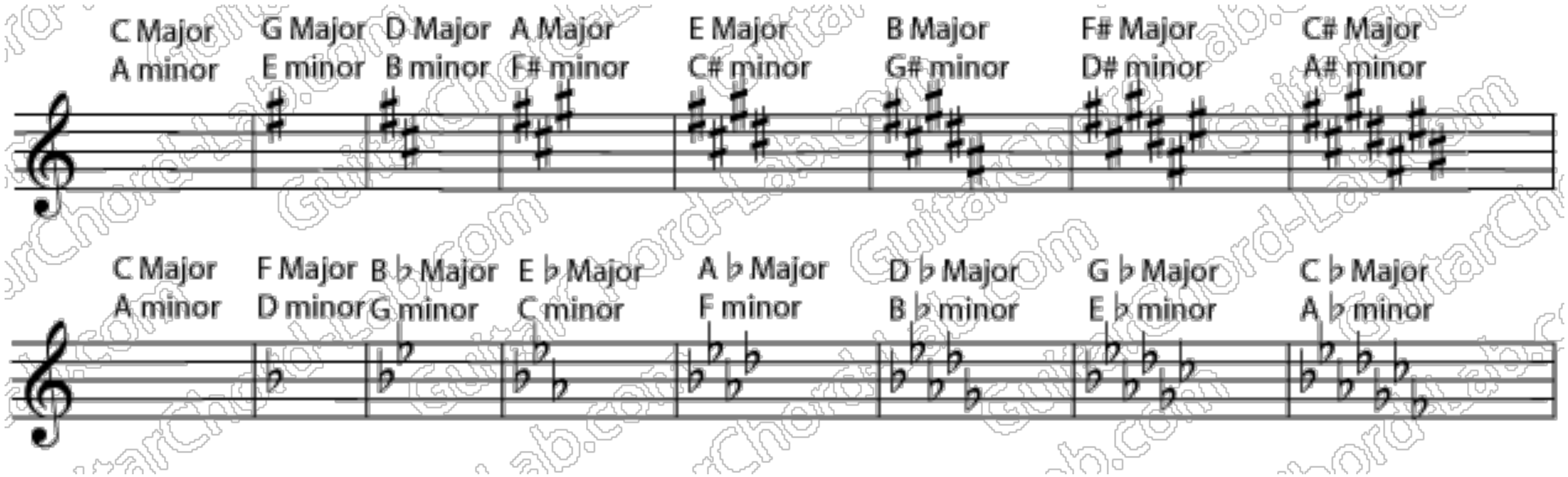
1=C　4/4



More about
Numbered Musical Notation

音　符		簡　譜 （以 Do 爲例）	休　止　符		簡　譜
全音符	𝅝	1 – – –	全休止符	𝄻	0　0　0　0
二分音符	𝅗𝅥	1 –	二分休止符	𝄼	0　0
四分音符	𝅘𝅥	1	四分休止符	𝄽	0
八分音符	𝅘𝅥𝅮	$\underline{1}$	八分休止符	𝄾	$\underline{0}$
十六分音符	𝅘𝅥𝅯	$\underline{\underline{1}}$	十六分休止符	𝄿	$\underline{\underline{0}}$

附 點 音 符		簡　譜 （以 Do 爲例）	附 點 休 止 符		簡　譜
附點全音符	𝅝.	$\overset{\frown}{1\ -\ -\ -\ 1}\ -$	附點全休止符	𝄻.	000000
附點二分音符	𝅗𝅥.	1 – –	附點二分休止符	𝄼.	000
附點四分音符	𝅘𝅥.	1 ·	附點四分休止符	𝄽.	0 ·
附點八分音符	𝅘𝅥𝅮.	$\underline{1}$ ·	附點八分休止符	𝄾.	$\underline{0}$ ·
附點十六分音符	𝅘𝅥𝅯.	$\underline{\underline{1}}$ ·	附點十六分休止符	𝄿.	$\underline{\underline{0}}$ ·



Some more interested facts about video game music…

- 好和弦 NiceChord: https://youtu.be/lAEIrown7GI
  - 為什麼超級瑪莉會聽起來像是超級瑪莉？