Faculty of Engineering and Technology
Department of Electrical and Computer Engineering

ENCS 4110
COMPUTER DESIGN LABORATORY
Report #2
Experiment #7

“Generating Music using 8254 PIT on PC”

Prepared by:

Nsreen Tawafsha – 1182319

Supervised by:

Dr. Ahmad Afaneh

Teacher Assistant:

Eng. Heba Qadah

Section: 4

Date: 15 April 2021

1. Abstract

The aim of this experiment is to examine the PPI (programmable peripheral interface) and its various operating modes. It also aims to comprehend PIT (Programmable Interval Timer) and its various modes, as well as learn how to link it to a computer and how to use it to produce sounds with different frequencies.

Table of Content

List of Figures

List of Tables

2. Theory

2.1 PPI 8255A:-

PPI 8255 is a general purpose programmable I/O device designed to interface the CPU with its outside world such as ADC, DAC, keyboard etc. We can program it according to the given condition. It can be used with almost any microprocessor.

It consists of three 8-bit bidirectional I/O ports i.e. PORT A, PORT B and PORT C. We can assign different ports as input or output functions.

✓ Ports of 8255A:

8255A has three ports, i.e., PORT A, PORT B, and PORT C.

- **Port A** contains one 8-bit output latch/buffer and one 8-bit input buffer.
- **Port B** is similar to PORT A.
- **Port C** can be split into two parts, i.e. PORT C lower (PC0-PC3) and PORT C upper (PC7-PC4) by the control word.

These three ports are further divided into two groups, i.e. Group A includes PORT A and upper PORT C. Group B includes PORT B and lower PORT C. These two groups can be programmed in three different modes, i.e. the first mode is named as mode 0, the second mode is named as Mode 1 and the third mode is named as Mode 2.

✓ Operating Modes:

8255A has three different operating modes:

- **Mode 0:** In this mode, Port A and B is used as two 8-bit ports and Port C as two 4-bit ports. Each port can be programmed in either input mode or output mode where outputs are latched and inputs are not latched. Ports do not have interrupt capability.

- **Mode 1:** In this mode, Port A and B is used as 8-bit I/O ports. They can be configured as either input or output ports. Each port uses three lines from port C as handshake signals. Inputs and outputs are latched.

- **Mode 2:** In this mode, Port A can be configured as the bidirectional port and Port B either in Mode 0 or Mode 1. Port A uses five signals from Port C as handshake signals for data transfer. The remaining three signals from Port C can be used either as simple I/O or as handshake for port B.

✓ 8255 Architecture:

PPI block diagram is shown in **Figure 2.1.1** below.

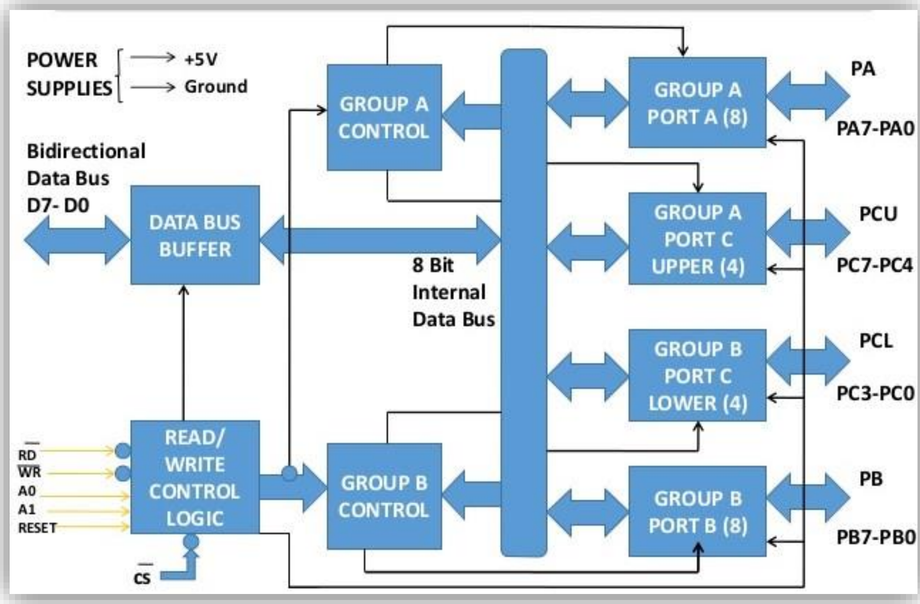

Figure 2.1.1: PPI 8255A block diagram

✓ 8255 Configuration using command byte:

Figure 2.1.1 shown PPI Configuration.

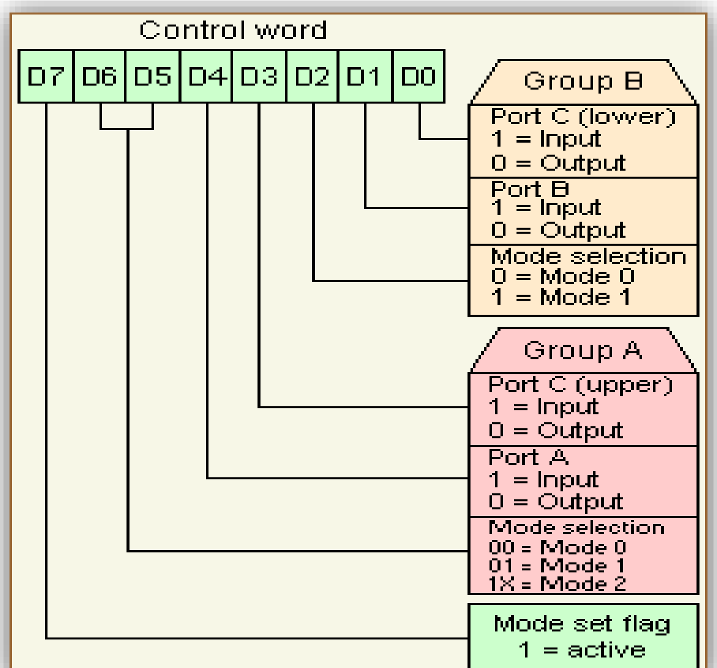

Figure 2.1.1: PPI Configuration

2.2 PIT 8253:-

The Intel 8253 Programmable Interval Timers (PTIs) designed for microprocessors to perform timing and counting functions using three 16-bit registers. Each counter has 2 input pins, i.e. Clock & Gate, and 1 pin for "OUT" output. To operate a counter, a 16-bit count is loaded in its register. On command, it begins to decrement the count until it reaches 0, then it generates a pulse that can be used to interrupt the CPU. **Figure 2.2.1 shows** 8253 Diagram, there are three counters, a data bus buffer, Read/Write control logic, and a control register. Each counter has two input signals - CLOCK & GATE, and one output signal - OUT.

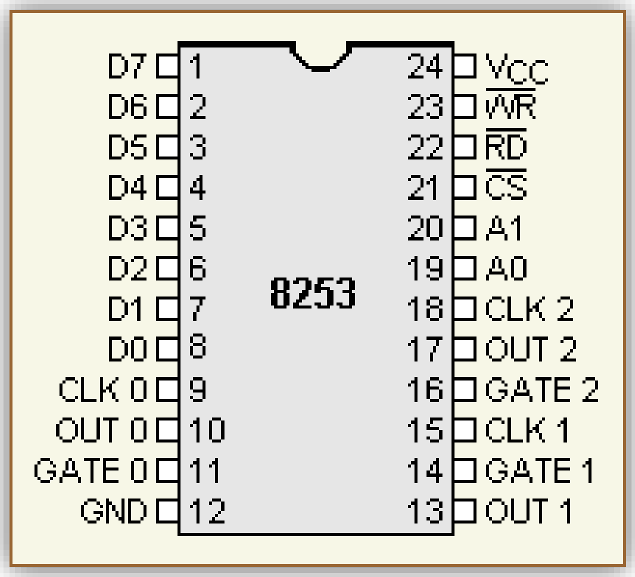

Figure 2.2.1: PIT 8253 Diagram

✓ **Operating Modes:**

8253 has six different operating modes:

- **Mode 0:** Interrupt on Terminal Count.
- **Mode 1:** Hardware Retriggerable One-Shot.
- **Mode 2:** Rate Generator.
- **Mode 3:** Square Wave Generator.
- **Mode 4:** Software Triggered Strobe.
- **Mode 5:** Hardware Retriggerable Strobe.

Table 2.2.2 shows the different uses of the 8253 gate input pin. Each mode of operation for the counter has a different use for the GATE input pin.

Table 2.2.2: PIT 8253 modes

Signal Status	Low or going low	Rising	High
Mode			
0	Disables counting	- -	Enables counting
1	- -	1) Initiates counting 2) Resets output after next clock	- -
2	1) Disables counting 2) Sets output immediately high	1) Reloads counter 2) Initiates counting	Enables counting
3	1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
4	Disables counting	- -	Enables counting
5	- -	Initiates counting	- -

✓ 8253 Architecture

Figure 2.2.3 shows the PIT 8253 Architecture.

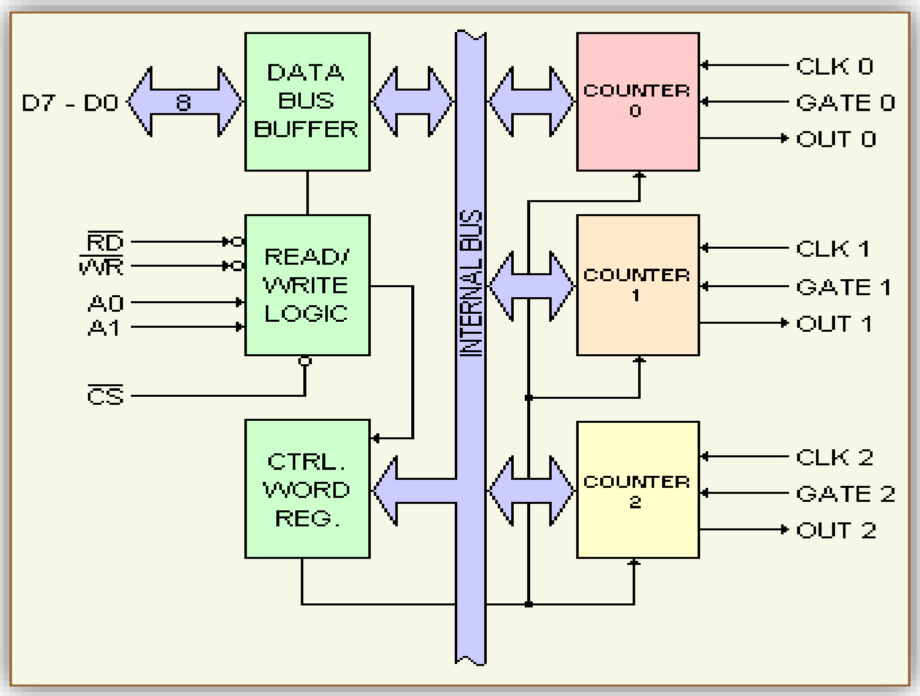

Figure 2.2.3: PIT 8253 Architecture

✓ 8253 Configuration using command byte:

Figure 2.2.4 shows the PIT 8253 Configuration.

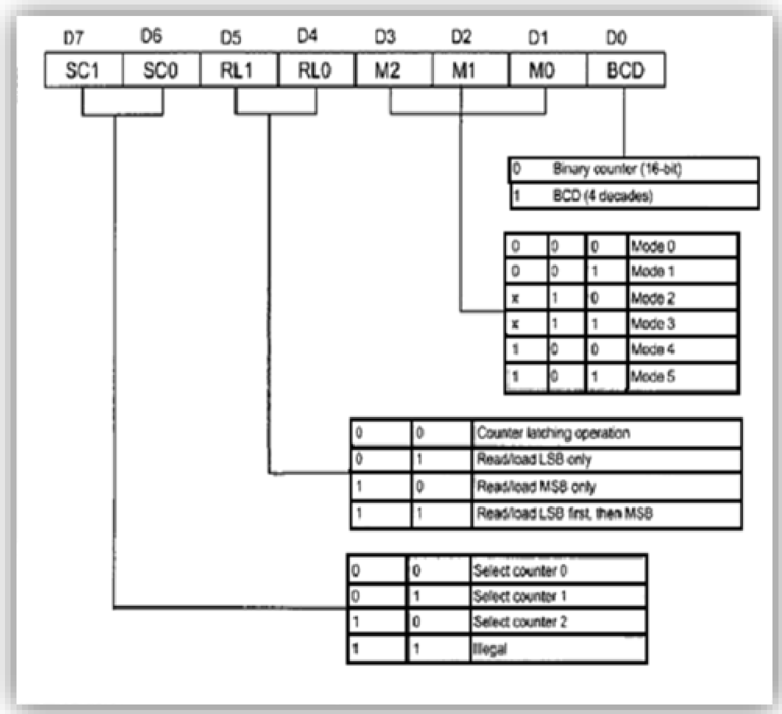

Figure 2.2.4: PIT 8253 Configuration

3. Procedure and Discussion

We must first specify the addresses for the counters and the PIT command register before we can begin the experiment's tasks. We do that using 3-8 decoder shown in **Figure 3.3.1**

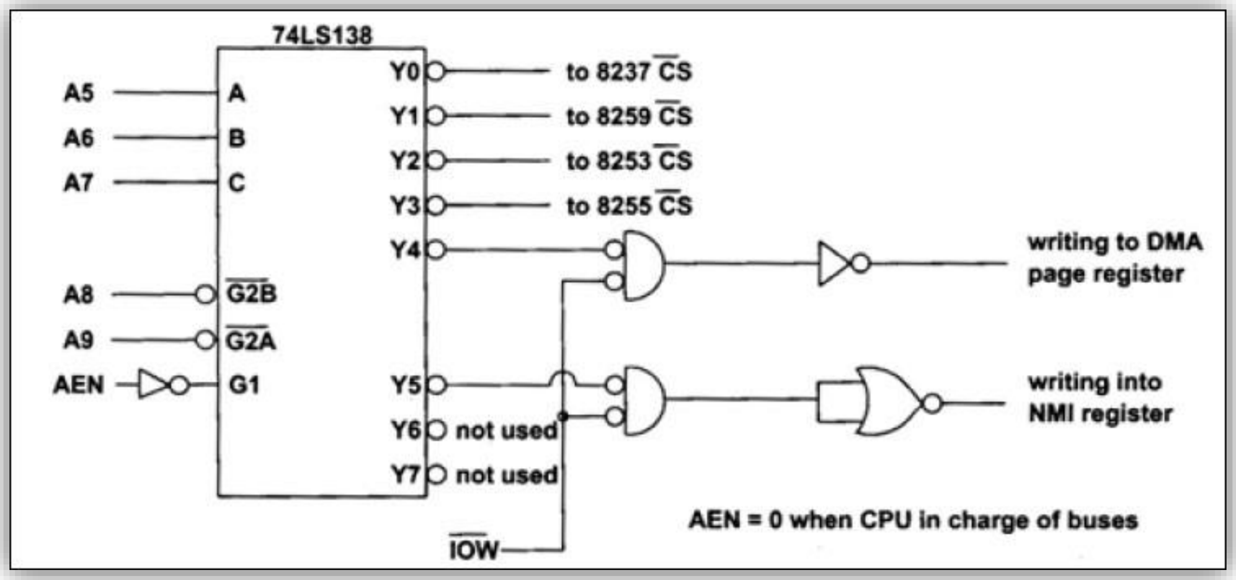

Figure 3.3.1: 3-8 Decoder

Figure 3.3.1 shows a 3-8 Decoder. Y2 is connected to the chip select of PIT and Y3 is connected to the chip select of PPI. Then, A7, A6, A5 will determine the base address for each of them as follows:

A7	A6	A5	A4	A3	A2	A1	A0	
0	1	1	0	0	0	0	0	→ 60h for PPI
0	1	0	0	0	0	0	0	→ 40h for PIT

So the port addresses for the PPI and the PIT is as follows:

PPI	**PIT**
Port A: 60h	Counter 0: 40h
Port B: 61h	Counter 1: 41h
Port C: 62h	Counter 2: 42h
Command register: 63h	Command register: 43h

3.1 Part A: Generating Beep using debug

In this step, we will generate sounds with different frequencies using debug.
First, we need to program the command register using counter 2 to generate a square wave (mode 3). The control word will be as follows:
10 (counter 2) **11** (read LSB then MSB) **011** (mode 3) **0** (BCD) → 10110110 → B6h.
We will send this value to port 43h to set up the speaker.

To calculate the count value, we will use the following formula:

$$count\ value = \frac{freq_{in}}{freq_{out}}$$

The frequency is 10K → $count\ value = \frac{1.19\ MHz}{10\ KHz} = 119\ Hz = 77h$, this frequency will be sent to port 42h as LSB then MSB.
To generate the sound, a value of 0011 (enable speaker and enable counter) will be sent to port 61h and to stop it we will use 0000. **Figure 3.1.1** shows this representation.

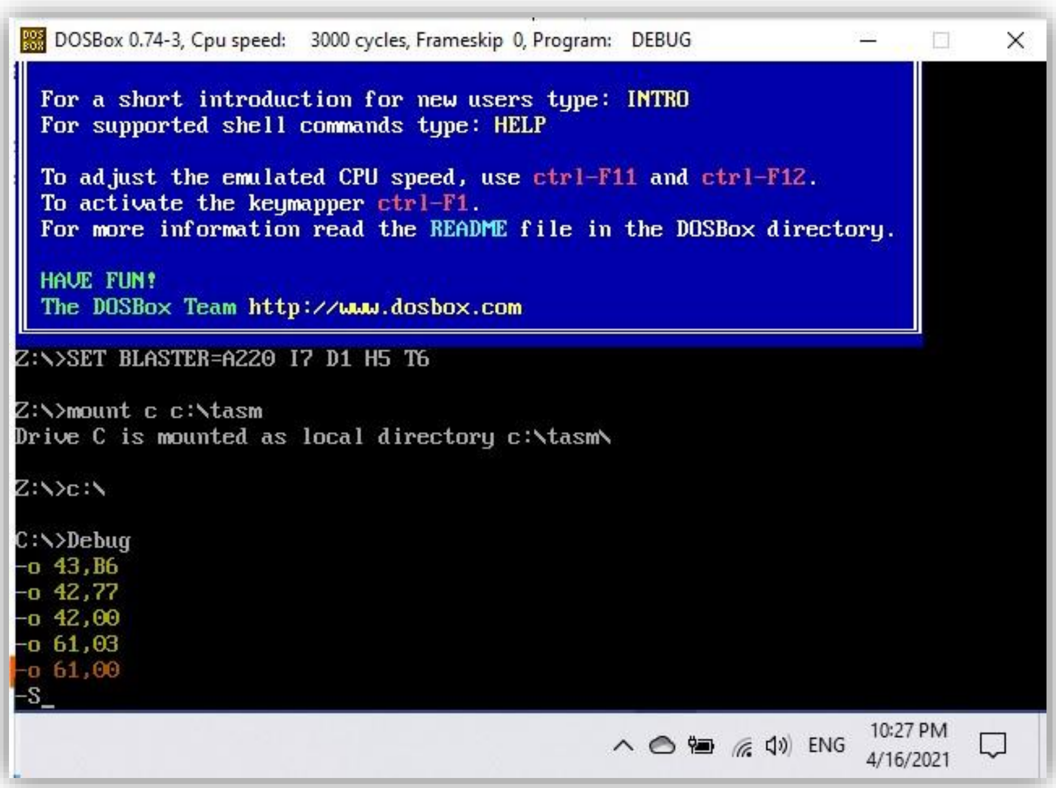

Figure 3.1.1: Generate Sound with 10 kHz

We should repeat the above steps to produce beep with frequency 1 KHz, 5 KHz, 15 KHz

- 1 KHz:

$$count\ value = \frac{1.19\ MHz}{1\ KHz} = 1190\ Hz = 4A6h$$

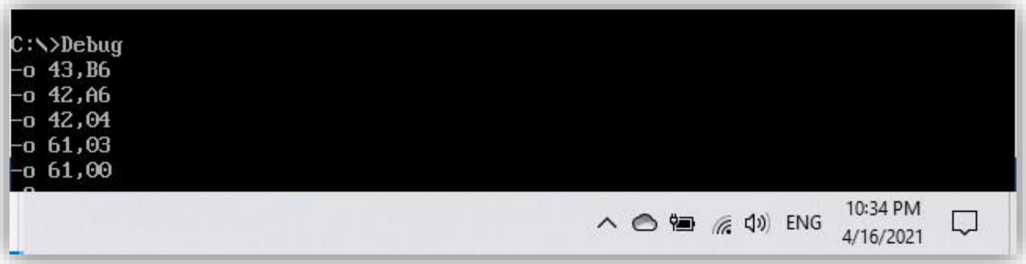

Figure 3.1.2: Generate Sound with 1 kHz

- 5 KHz:

$$count\ value = \frac{1.19\ MHz}{5\ KHz} = 238\ Hz = 0EEh$$

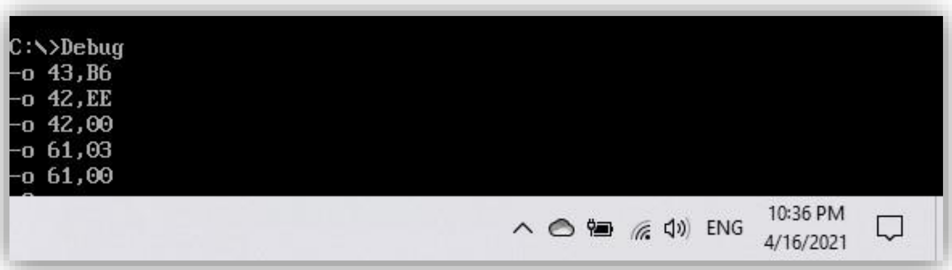

Figure 3.1.3: Generate Sound with 5 kHz

- 15 KHz:

$$count\ value = \frac{1.19\ MHz}{15\ KHz} = 79.33\ Hz = 4Fh$$

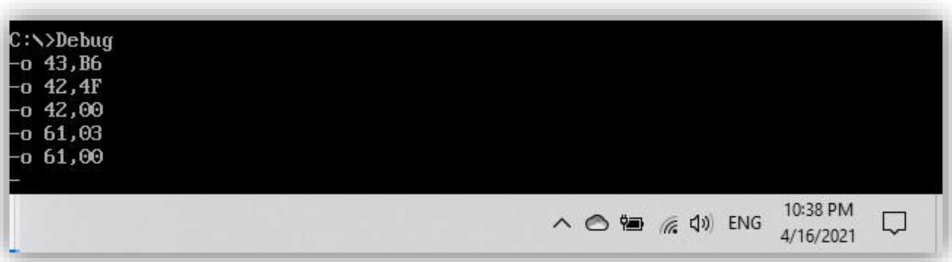

Figure 3.1.4: Generate Sound with 15 kHz

3.2 Part B: Using TASM to produce beep sounds

The code written at Appendix A used to generate a sound with frequency 200 Hz

In this code, to make a tone, we must first initialize the command register, and then submit the frequency value as LSB, then MSB to counter 2. To activate the tone, we will create an exe file with TASM and TLINK. **Figure 3.2.1** shows this representation.

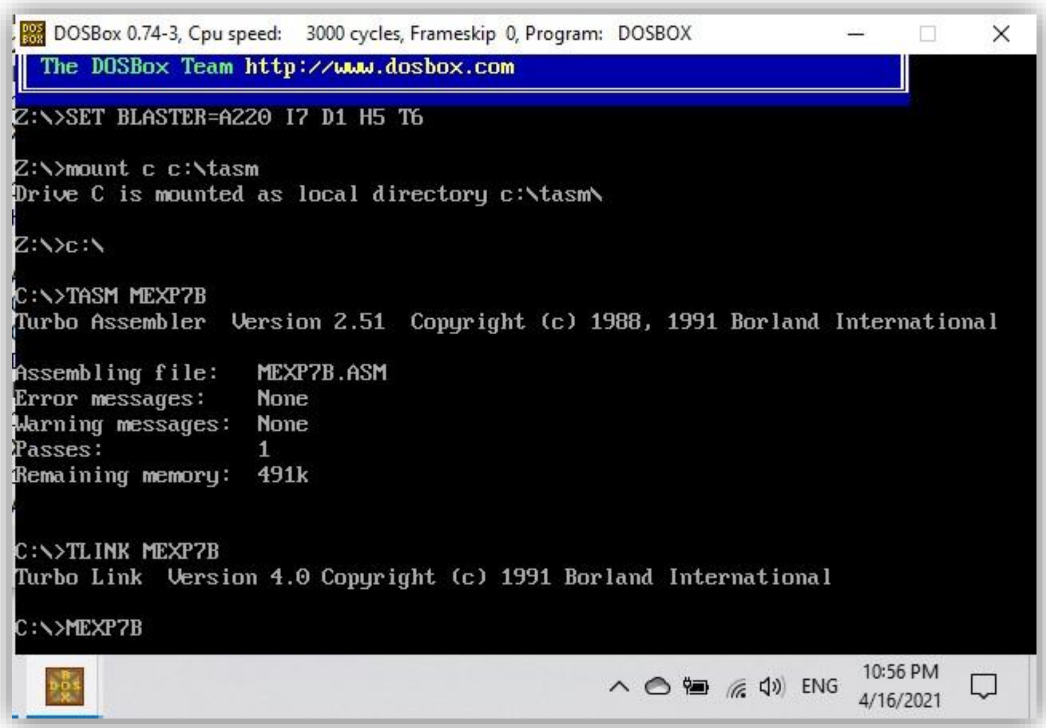

Figure 3.2.1: generate sound part B

- Change the code to produce a beep sound of 3 KHz for 5 seconds

$$count\ value = \frac{1.19\ MHz}{3\ KHz} = 396.6\ Hz = 18Ch$$

.DATA
COUNT EQU 18Ch
T EQU 5000

- Change the code to produce a beep sound of 12 KHz for 2 seconds

$$count\ value = \frac{1.19\ MHz}{12\ KHz} = 99\ Hz = 63h$$

.DATA
COUNT EQU 63h
T EQU 2000

3.3 Part C: Generate Music on your PC

In this part, we will write Assembly code to play “Happy Birthday “using the code in Appendix B.

In the code, we defined a MACRO with name tone to generate each sound of 'Happy Birthday'. It will takes the frequency and the duration of the sound, initialize the command register and then send the frequency as LSB and MSB to the counter to turn on the sound for the specific duration. **Figure 3.3.1** illustrates this representation.

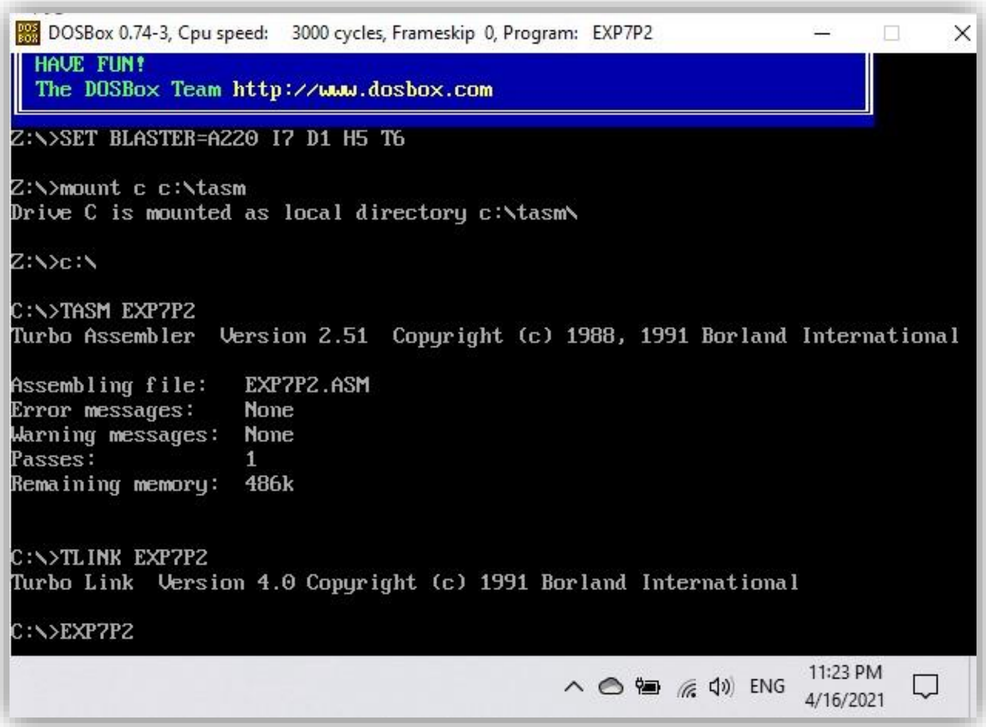

Figure 3.3.1: generate “Happy Birthday” sound part C

3.4 Part D: Using Keyboard as Piano keys

In this task, we will use the keyboard keys to generate different tones by mapping the letters on the keyboard to different tones.

In this part, the code at Appendix C will generate different sounds based on the pressed letter on the keyboard. Each letter is mapped to a specific frequency and when the letter is pressed, this value of frequency will be sent to the MACRO tone that will initialize the command register and generate the sound. **Figure 3.4.1** illustrates this representation.

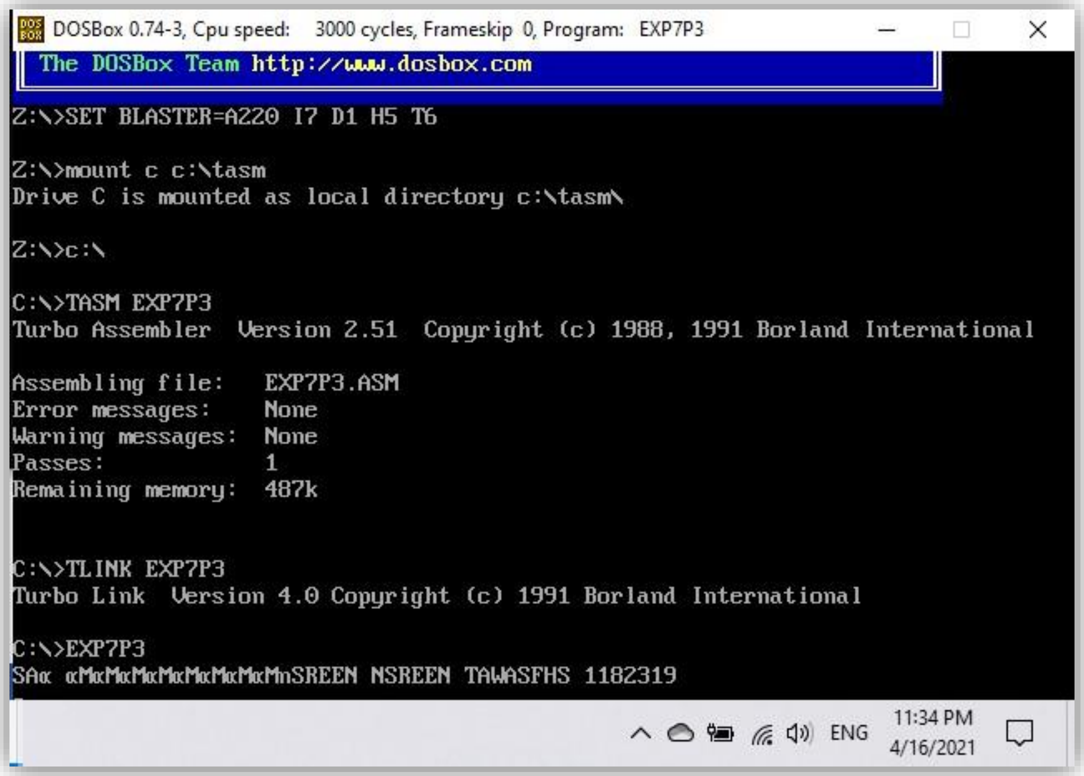

Figure 3.4.1: Using Keyboard as Piano keys

4. Conclusion

In this experiment, we learned how to the PIT works and how we can program it to generate different sounds with different frequencies.

5. References

[1] https://www.tutorialspoint.com/microprocessor/microprocessor_intel_8255a_programmable_peripheral_interface.htm Access Date: 16-4-2021 at 2:00PM.

[2] http://discipline.elcom.pub.ro/amp2/curs/8253.htm Access Date: 16-4-2021 at 3:00PM

[3] Lab Manual.

6. Appendix

6.1 Appendix A:

```
.MODEL SMALL
.STACK 1000H

.DATA
COUNT EQU 200D
T EQU 500

.CODE
START:
 MOV AL,0B6H
 OUT 43H,AL
 MOV AX,COUNT
 OUT 42H,AL
 MOV AL,AH
 OUT 42H,AL
 MOV AL, 00000011B
 OUT 61H,AL
 MOV CX,T

 DELAY1:
  PUSH CX
  MOV CX,20000
  DELAY2:
    LOOP DELAY2
    POP CX
 LOOP DELAY1
 MOV AL,00000000B;DISABLE GATE
 OUT 61H,AL
 MOV AX,4C00H
 INT 21H
END START
```

6.2 Appendix B:

```
.model small
.stack 1000h
.data
t equ 20
.code
tone macro div,dur
mov al,0b6h
out 43h,al

mov ax,div
out 42h,al

mov al,ah
out 42h,al

mov al,00000011b
out 61h,al

mov cx,dur
call delay1

mov al,00000000b
out 61h,al

call delay2
endm

.startup
tone 4553,t ; hap (c4)
tone 4553,t ; py (c4)
tone 4057,2*t ; birth (d4)
tone 4553,2*t;c4
tone 6409,2*t;f4
tone 3606,4*t;e4
tone 4553,2*t;c4
tone 4553,2*t;c4
tone 4057,2*t;d4
tone 4553,2*t;c4
tone 3035,2*t;g4
tone 3409,4*t;f4
tone 4553,t;c4
tone 4553,t;c4
tone 2275,2*t;c5
tone 2704,2*t;a4
tone 3409,2*t;f4
tone 3608,2*t;e4
tone 4057,6*t;d4
tone 2553,t;b4b
tone 2553,t;b4b
tone 2704,2*t;a4
```

```
tone 3409,2*t;f4
tone 3035,2*t;g4
tone 3409,4*t;f4
; Continue your code here……
mov ah,4ch
int 21h

delay1 proc near
d1:
push cx
mov cx,38000
d2:
loop d2
pop cx
loop d1
ret
delay1 endp
delay2 proc near
mov cx,65000
d3:
loop d3
ret
delay2 endp
end
```

6.3 Appendix C:

```
.model small
.stack 1000h
.data
t equ 10
.code
tone macro div,dur
mov al,0b6h
out 43h,al
mov ax,div
out 42h,al
mov al,ah
out 42h,al
mov al,00000011b
out 61h,al
mov cx,dur
call delay1
mov al,00000000b
out 61h,al
call delay2
endm
.startup

lab:
mov ah,1h
int 21h
cmp al,'a'
jz A
cmp al,'b'
jz B
cmp al,'c'
jz Ce
cmp al,'d'
jz n5
jmp s5
n5:
jmp D
s5:
cmp al,'e'
jz n1
jmp s1
n1:
jmp E
s1:
cmp al,'f'
jz n2
jmp s2
n2:
jmp F
s2:
cmp al,'g'
```

```
jz n3
jmp s3
n3:
jmp G
s3:
cmp al,'h'

jz n4
jmp s4
n4:
jmp H
s4:
A:
tone 2704,t;a4
jmp lab
B:
tone 2553,t;b4b
jmp lab
Ce:
tone 4553,t;c4
jmp lab
D:
tone 4057,t;d4
jmp lab
E:
tone 3608,t;e4
jmp lab
F:
tone 3409,t;f4
jmp lab
G:
tone 3035,t;g4
jmp lab
H:
tone 2275,t;c5
jmp lab
; Continue your code here……
q:
mov ah,4ch
int 21h
delay1 proc near
d1:
push cx
mov cx,38000
d2:
loop d2
pop cx
loop d1
ret
delay1 endp
delay2 proc near
mov cx,65000
```

```
d3:
loop d3
ret
delay2 endp
end
```