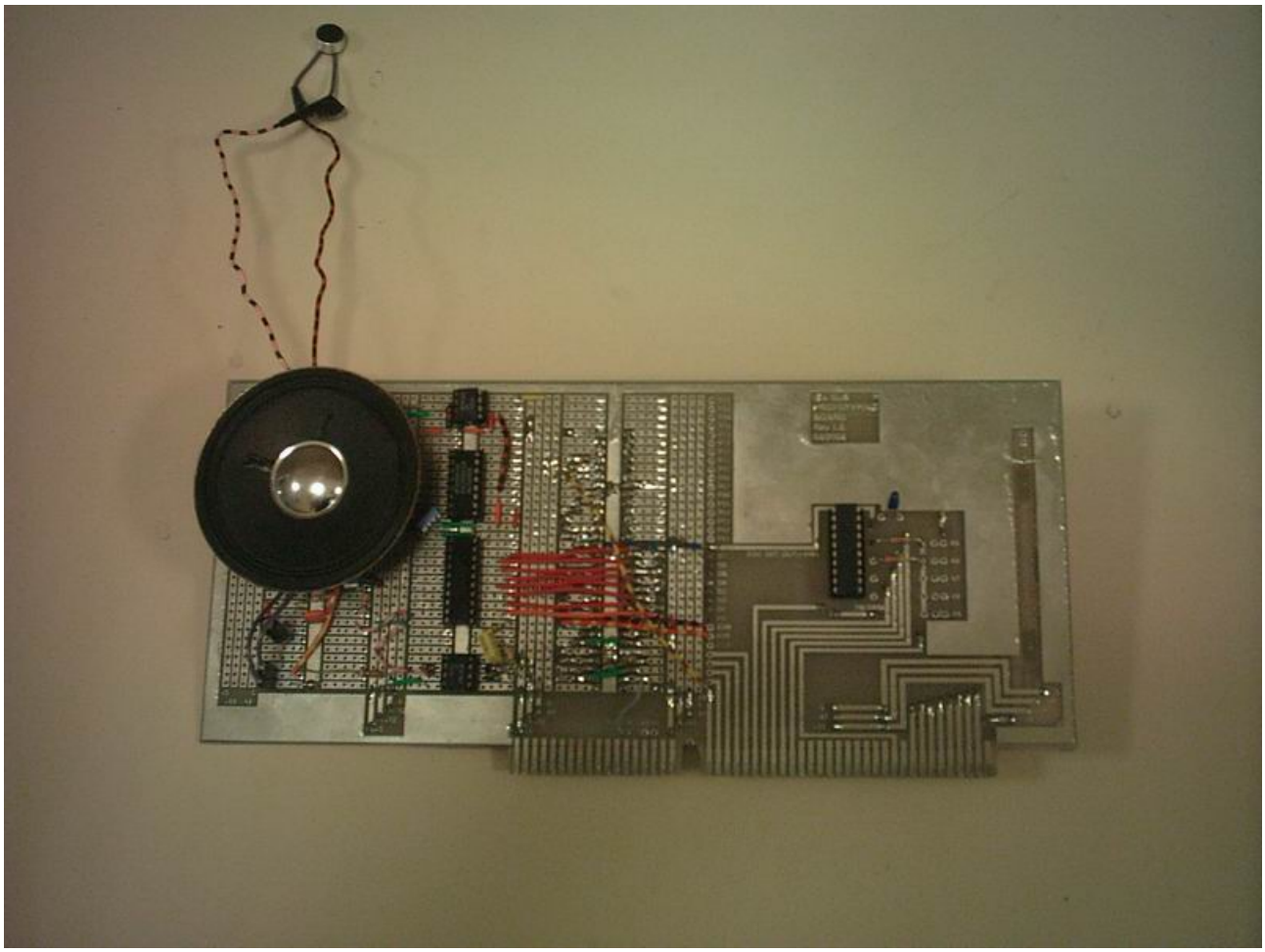
EL308 – Microprocessor Based System Design

Term Project'04

“8-Bit Custom Made SOUNDCARD”



Final Report

by
M.Onur SARKAN & Erman ENGİN

Outline:

1- Introduction:

In the digital era, every data, sound, video are stored and processed digitally. Aim of our project was to build a 8-Bit custom made Soundcard by using simple components like ADC/DAC chips, operational amplifiers, resistances, capacitors, etc. Our soundcard works by ISA Bus in any computer which have a X86 families microprocessor. Our soundcard has two main jobs: Sound recording and Sound playing. In the sound recording case, soundcard takes analog sound signal from capacitive microphone, and converts this sound signal to 8 bit digital format to store in memory of computer. In the sound playing case, soundcard takes 8 bit format digital signal from specific memory location, and converts this digital signal to analog sound format to play by speaker. Following figure is belong to overall system hardware, which includes both sound recording and playing hardware

.

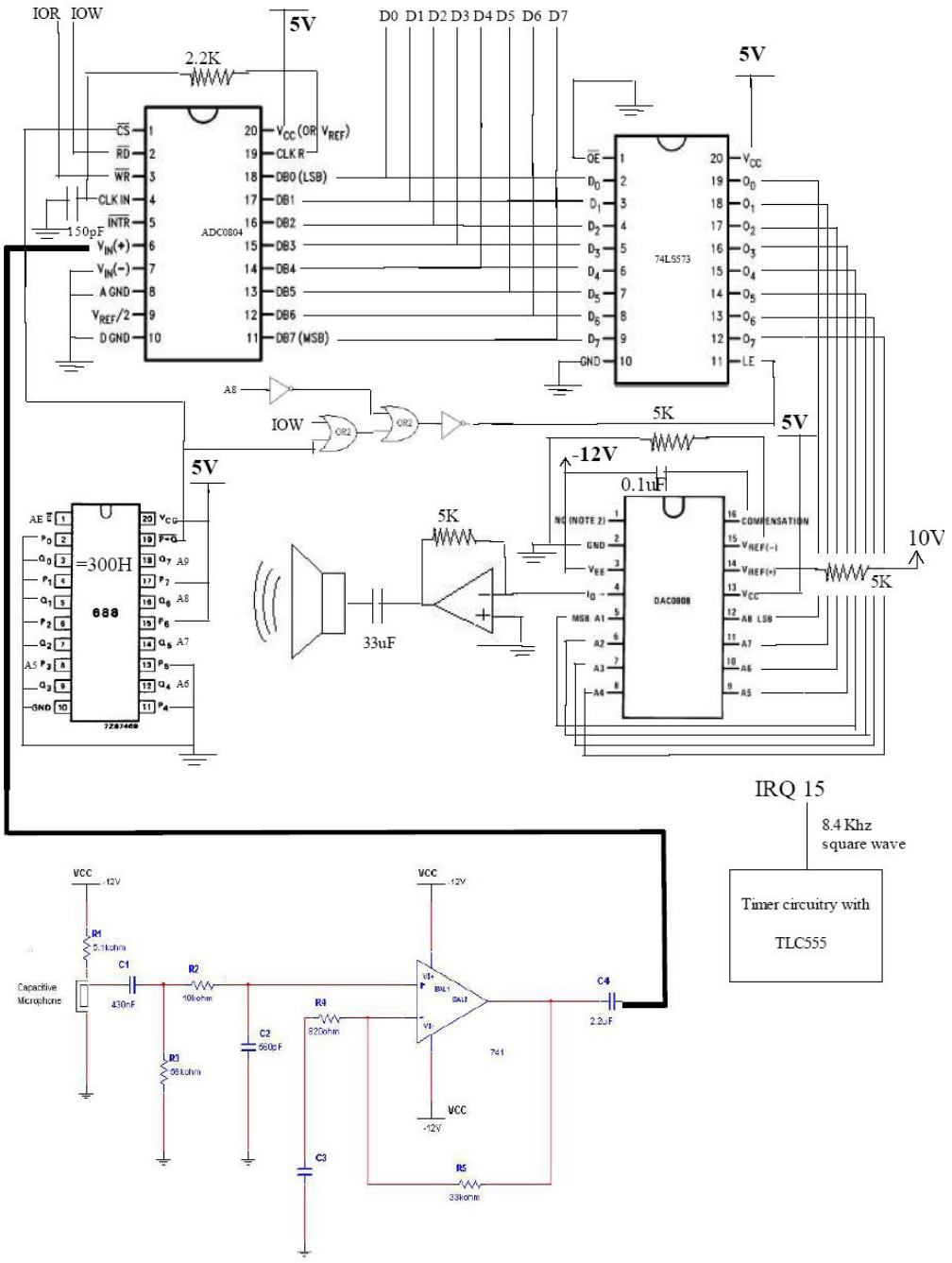


Figure 1(Overall Circuit)

2- Sound Recording System:

Our Sound Recording system, there are two main subsystems. First subsystem analog signal amplification, second part is Analog to Digital Conversion.

a- Analog Signal Amplification:

Soundcard takes analog sound signal from capacitive microphone, but output signal of capacitive microphone is about peak to peak 20-30 mV. Because of this low-level signal voltage, we amplify the sound signal by following analog amplifier circuitry in figure 2. Gain of this circuitry is about 200 mV.

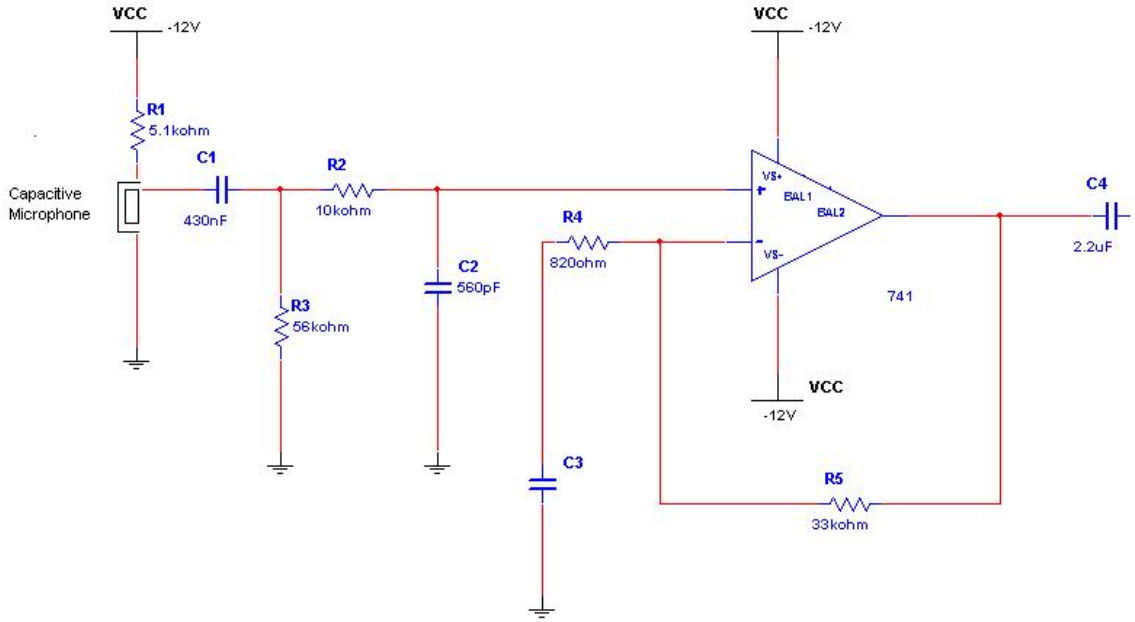


Figure 2(Analog Amplifier Circuitry)

We tested our analog amplifier circuitry by oscilloscope. Following figure is output of amplifier circuitry when we are whistling. You can see smooth amplified sinusoidal signal due to the human whistling in figure 3.

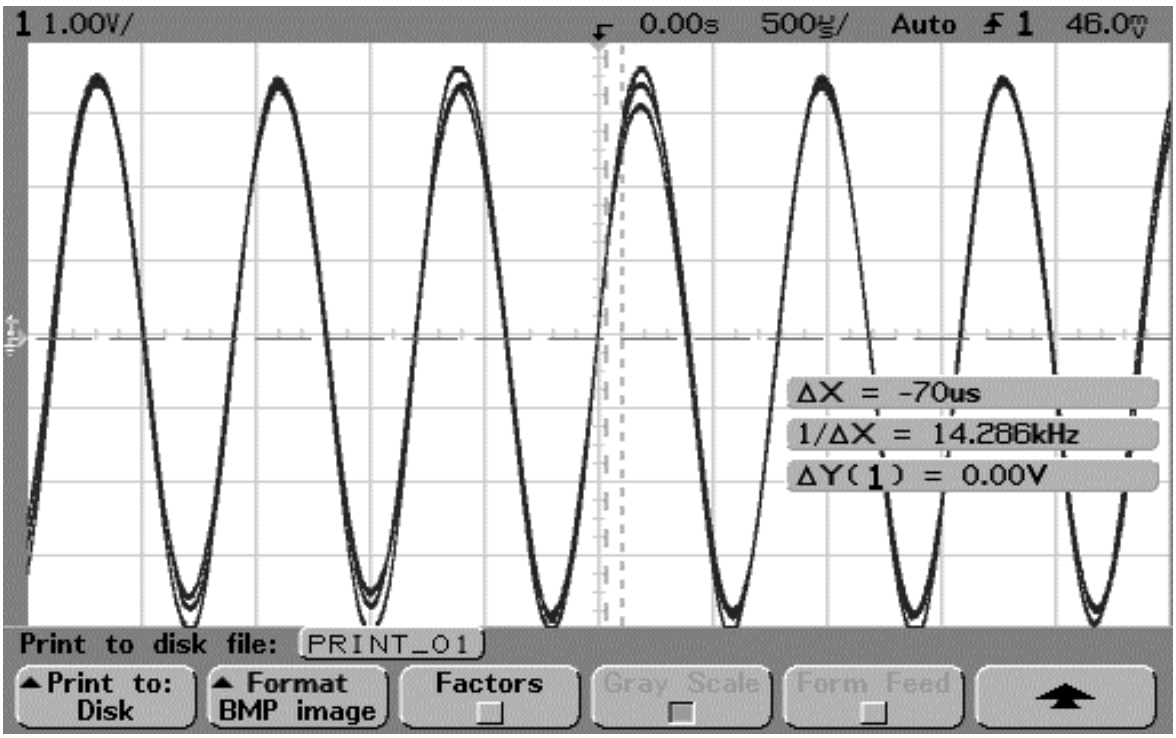


Figure 3(Amplified Human whistle signal)

b- Analog to Digital Conversion & Writing Binary Sound File:

In this part, we apply Analog to Digital conversion on amplified sound signal to be able to record in computer memory. By 8-bit ADC, we take periodic samples (in our case 8400 samples per second) from analog input (0-5 Volt range), and represent these samples by 256 discrete voltage levels. We used Texas Instrument ADC 0804 chip.

After amplification of sound signal, we add 2,5 Volt DC voltage to our signal, and we send this signal to ADC circuitry as an input. ADC chip is used to convert an analog electrical signal swinging between 0-5Volts to an 8-bit data. 8000 samples per second is to be taken from the analog input. In order to provide timing the ADC0804 chip has pins 4 and 19 for adjusting the conversion time. Simple capacitor and resistor connection to these pins is sufficient to provide that. Following figure is belong to ADC circuitry.

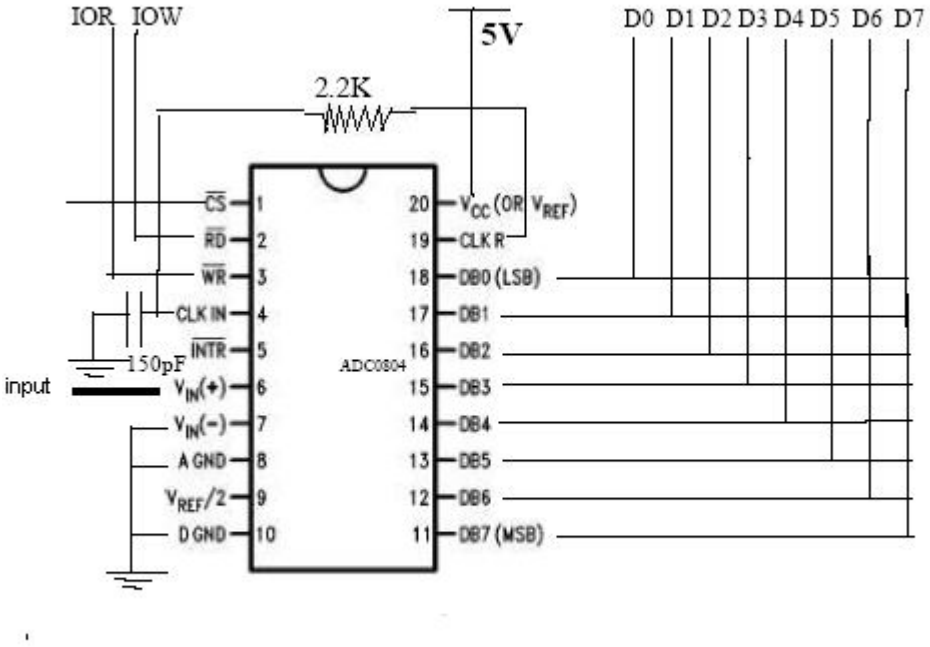


Figure 4(ADC Circuitry)

In order to provide timing, we used TLC555 Timer Chip. This timer chip sends 8.4 kHz square signals to IRQ 15 pin. When processor takes this interrupt signal, processor sends "start conversion" signal to ADC chip. The data produced by the ADC will be read from the data bus simply by sending the read signal to the ADC chip. By timer chip, we can take 8400 samples per second from ADC Chip. At the end of the ADC stage, sound signal is recorded into memory.

3- Sound Playing System:

In this part of soundcard, we use TLC555 timer chip again in order to reach same time scaling. For each interrupt, digital sound data is read from specific memory location. These bit streams are sent to D-Latched. We just enable and disable D-Latched. DAC chip read data from this D-Latched. In the DAC chip, 8-bit data is converted to –5-0 volt analog output. We invert and scale this signal, and send it speaker. Following graph is overall circuitry of DAC.

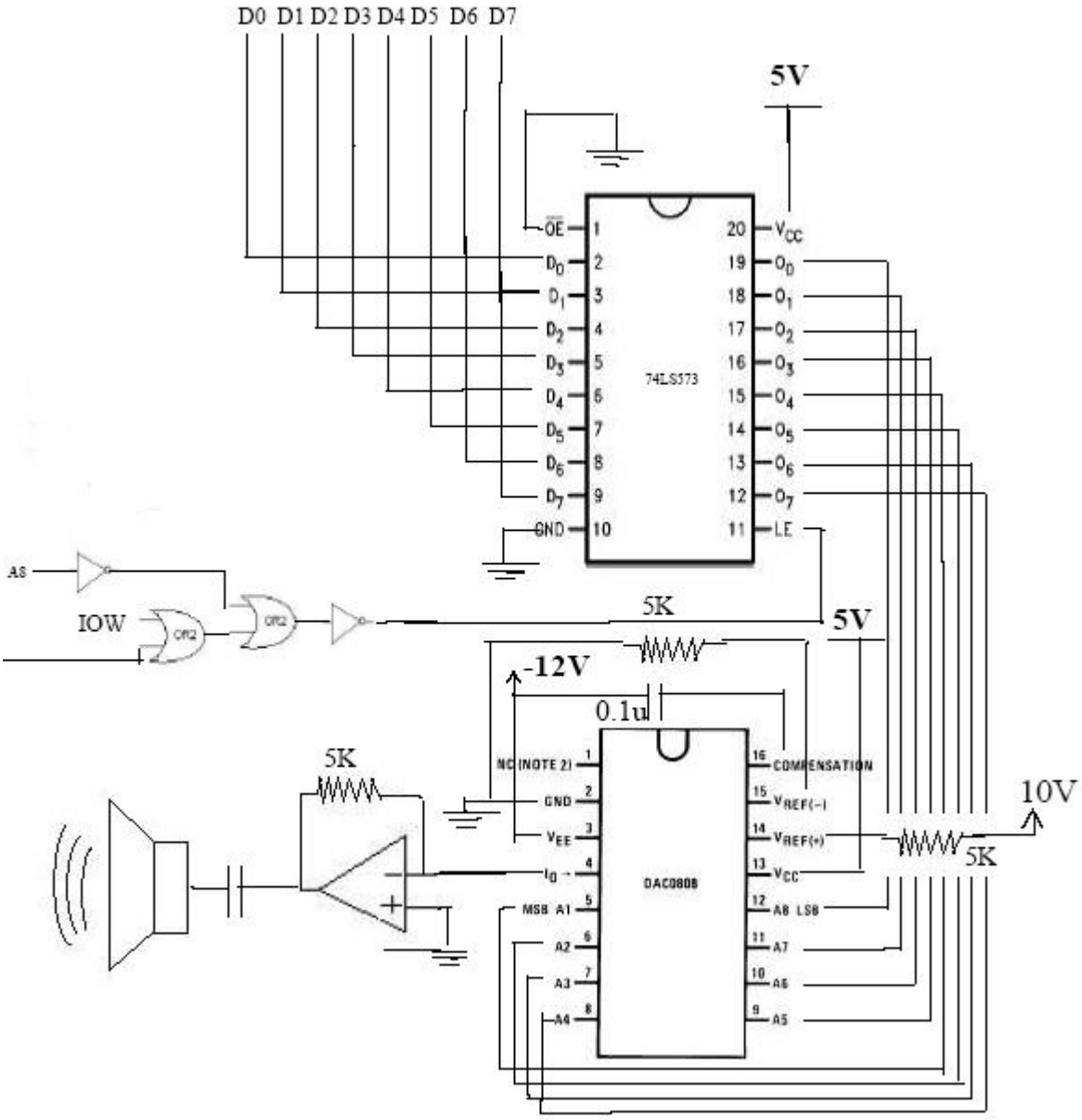


Figure 5(Overall DAC Circuitry)

We tested our Sound Playing System manually. When we read record of 1 kHz Analog sinusoidal signal, we took following output from output of DAC circuitry under 8 kHz sampling rate.

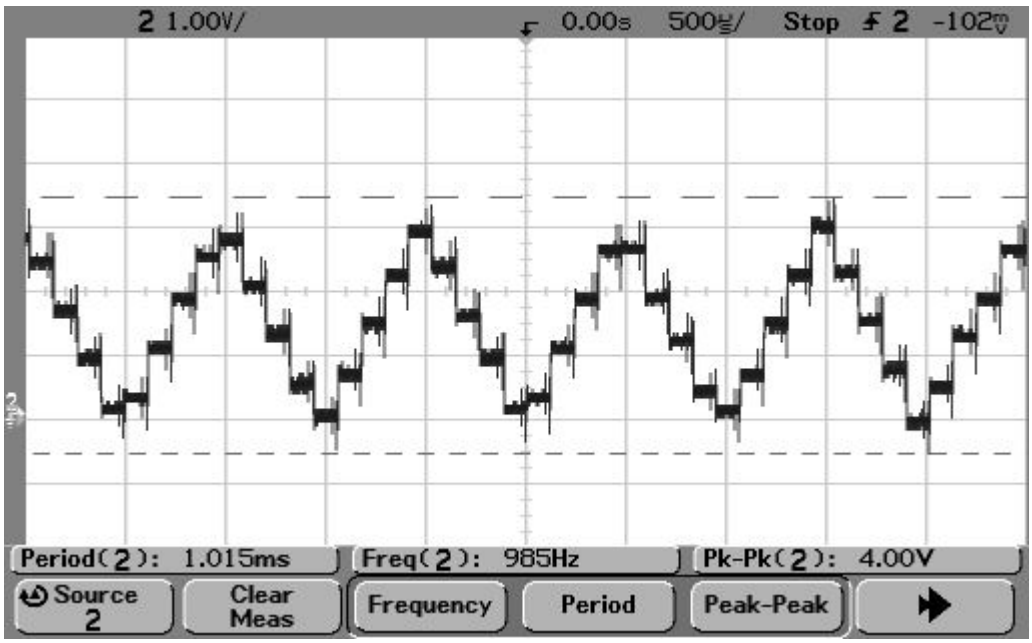


Figure 6(DAC Output of 1 kHz sinusoidal signal with 8 kHz Sampling Rate)

4- Software:

There are 3 separate software named ISR, Record and Play:

ISR-The Interrupt service routine:

The job of the ISR is:

1. Enable IRQ_10 port of the 8259 Chip where the **timer circuit** is directly connected.
    1. Hardware interrupt routine sets the "Interrupt received Flag" and dies out.
2. Install Software Interrupt "int 255" named Status
    1. Status routine moves the current status of the "Interrupt received Flag" to register AL, then resets "Interrupt received Flag" and dies out.

Record:

Record program takes up to 25 characters of file name with its function name.(e.g. Record myVoice.mos).

Then:

- A file with this name is made under "C:\"
- Enters the "writeloop"
    - writeloop checks the status of the keyboard and the the status of "Interrupt received Flag" by softint Status (int 255).
        - If the user presses "S", the program will end
        - If the user presses "P", the program pauses and waits until another "P" from the user in order to resume or a "S" from the user in order to end.
        - If "Interrupt received Flag" is set, takes the data from ADC chip and writes it to the file.

Play:

Record program takes up to 25 characters of file name with its function name.(e.g. Play myVoice.mos).

Then:

- The file with the name of user's input is opened if it exists, an error is given if it doesn't exist or cannot be opened.
- Enters the “readloop”
  - readloop checks the status of the keyboard and the the status of “Interrupt received Flag” by softint Status (int 255).
    - If the user presses “S”, the program will end
    - If the user presses “P”, the program pauses and waits until another “P” from the user in order to resume or a “S” from the user in order to end.
    - If “Interrupt received Flag” is set, sends the data in the variable “INPUT_CHAR” to the DAC chip and takes the next data from the file to “INPUT_CHAR”.

5- Conclusion:

Finally, we implemented working 8-bit soundcard, which has record, play, pause, and resume capabilities. There is some noise at speaker but it is normal because of quantization noise and additive white gaussian noise.

6- Appendix:

ISR.ASM:

```
;**********MACROS****************

STRING MACRO DATA1
        MOV AH,09H
        MOV DX,OFFSET DATA1
        INT 21H
ENDM

;*********************************

ADCPORT EQU 301H
DACPORT EQU 300H

INTDISMASTER EQU 20H
INTENMASTER EQU 21H
INTDISSLAVE EQU 0A0H
INTENSLAVE EQU 0A1H
EOIMASTER EQU 62H
EOISLAVE EQU 67H

        .MODEL SMALL
        .DATA
  .CODE
Start:
      mov ax,@Data
```

```
        mov dx,ax
          cli                           ;Enable Interrupt 2 from 8259
;*************************************************************************
        mov dx,INTENMASTER
        in al,dx
        and al,11111011b
        out dx,al               ;Enable Interrupt 15 from Slave 8259
;*************************************************************************
        mov dx,INTENSLAVE
        in al, dx
        and al, 01111111b
        out dx, al

        sub ax,ax
        mov es,ax                       ;Install HardInt
;*************************************************************************
        mov ax,OFFSET HardInt
        mov es:[4*77h+0],ax
        mov ax,cs
        mov es:[4*77h+2],ax             ;Install Softint Status
;*************************************************************************
        mov ax,OFFSET Status
        mov es:[4*255+0],ax
        mov ax,cs
        mov es:[4*255+2],ax             ;Stay Resident and Terminate
;*************************************************************************
          mov ah,31h
        mov dx,OFFSET Last
        inc dx
          sti
        int 21h                         ;VARIABLES
;*************************************************************************
IntRecieved DB 0                        ;HardInt
;*************************************************************************

HardInt:
          push es
          push ds
          push ax
          push dx

          mov IntRecieved,1     ;Send End Of Interrupt to Master 8259
;---------------------------------------------------------------
          mov dx,INTDISMASTER
          mov al,EOIMASTER
          out dx,al             ;Send End Of Interrupt to Slave 8259
;---------------------------------------------------------------
          mov dx,INTDISSLAVE
          mov al,EOISLAVE
          out dx,al

          pop dx
```

```
        pop ax
        pop ds
        pop es
iret                                    ;Status
;*************************************************************************

Status:
        push es
        push ds

        mov al,IntRecieved
        mov IntRecieved,0

        pop ds
        pop es
iret                                    ;Terminate
;*************************************************************************
Last:
      END Start
```

RECORD.ASM

```
;**********MACROS****************

STRING MACRO DATA1
        MOV AH,09H
        MOV DX,OFFSET DATA1
        INT 21H
ENDM

;*********************************
;RECORD PROGRAM IS USED TO TAKE IN THE DATA CONVERTED BY THE ADC CHIP
ON PORT ADRESS
;301H AN DSTORE IT INSIDE THE DATA SEGMENT OF 64 KBYTES.

INTENMASTER EQU 21H
INTENSLAVE EQU 0A1H
ADCPORT EQU 300H

        .MODEL SMALL
        .DATA
                                        ;FILE VARIABLES
;*************************************************************************
FILE_NAME DB "c:\",25 DUP(?)
FILE_HANDLE DW 0                        ;FILE ERROR MESSAGES
;*************************************************************************
err_makefile DB "ERROR MAKING THE FILE!","$"
err_writefile DB "ERROR WRITING TO THE FILE!","$"
err_closefile DB "ERROR CLOSING THE FILE!","$"
;err_openfile DB "ERROR OPENING THE FILE!","$"
;err_readfile DB "ERROR READING THE FILE!","$"
;*************************************************************************
```

```
INPUT_CHAR DB "a"                       ;PROGRAM MESSAGES
;*************************************************************************
START_REQUEST DB "PRESS 'S' TO START RECORDING ",13,10,"$"
MSG_RECORDING DB "RECORDING STARTED. PRESS 'S' AGAIN TO STOP OR 'P' TO
PAUSE",13,10,"$"
MSG_PAUSE DB 10,"PAUSED. PRESS 'P' AGAIN TO RESUME OR 'S' TO STOP",13,10,"$"
MSG_RESUME DB "RESUMED",13,10,"$"               ;DEBUG VARIABLES
;*************************************************************************
MSG_DEBUG1 DB "DEBUG: DATA AVAILABLE? ",10,"$"
MSG_DEBUG2 DB "DEBUG: CHECKING KEY ",10,"$"
MSG_DEBUG3 DB "DEBUG: WRITING ",10,"$"
;*************************************************************************

        .CODE
MAIN:
        MOV AX,@DATA
        MOV DS,AX               ;Taking the file name
;*****************************************************************
        XOR CX,CX
        MOV CL,ES:[80H]
        DEC CL
        MOV SI,2
        MOV DI,3

Input_File_Name:
        mov bl,es:[80h][si]
        mov FILE_NAME[DI],bl
        inc si
        inc di
loop Input_File_Name
        mov FILE_NAME[DI],0
        inc di
        mov FILE_NAME[DI],"$"           ;Make the File named FILE_NAME
;*****************************************************************
        mov ah,3ch
        mov cx,0
        mov dx,offset FILE_NAME
        int 21h
jnc FMSUCCEEDED
        string err_makefile
        jmp endprog
FMSUCCEEDED:
        mov FILE_HANDLE,AX              ;Wait for start instruction
;*************************************************************************
        string START_REQUEST
startcommand:
        mov ah,07h
        int 21h
        cmp al,"S"
jne startcommand                ;Start writing to the file
;*************************************************************************
        string MSG_RECORDING
```

```
writeloop:                      ;cHECK KEYBOARD
;-----------------------------------------
        mov ah,0Bh
        int 21h
        cmp al,0
jz check_data
        mov ah,07h
        int 21h
        cmp al,"S"
jz close_file
        cmp al,"P"
jnz check_data
        string MSG_PAUSE
wait_command:
        mov ah,07h
        int 21h
        cmp al,"S"
jz close_file
        cmp al,"P"
jnz wait_command
        string MSG_RESUME
check_data:                     ;CHECK INTERRUPT
;-------------------------------------------------
;XXXXXXXXXXCHECK DATA AVAILIBILITYXXXXXXXXX
        int 255
        cmp al,0
jz end_of_writeloop
;XXXXIF AVAILABLE COPY IT TO INPUT_CHAR XXXXXXXX
        mov dx,ADCPORT
        in al,dx
        out dx,al

        mov INPUT_CHAR,al
        mov ah,40h
        mov bx,FILE_HANDLE
        mov cx,1
        mov dx,offset INPUT_CHAR
        int 21h

end_of_writeloop:
jmp writeloop                   ;Close the file
;*****************************************************************
close_file:
        mov ah,3eh
        mov bx,FILE_HANDLE
        int 21h
jnc closesucceeded
        string err_closefile
        jmp endprog
closesucceeded:                 ;Disable Interrupt on Master 8259
;*****************************************************************
      mov dx,INTENMASTER
```

```
        in al,dx
        or al,00000100b
        out dx,al                   ;Disable Interrupt 10 from Slave 8259
;*****************************************************************
        mov dx,INTENSLAVE
        in al, dx
        or al,10000000b
        out dx, al                  ;Terminate
;*****************************************************************
endprog:
        mov ah,4ch
        int 21h

END MAIN
```

PLAY.ASM

```
;**********MACROS*****************
STRING MACRO DATA1
        MOV AH,09H
        MOV DX,OFFSET DATA1
        INT 21H
ENDM
;*********************************
;PLAY PROGRAM IS USED TO READ THE DATA FROM A USER INPUT FILE AND
;SEND IT TO DAC IN PORT 300H
;*************************************************************************
**********


INTENMASTER EQU 21H
INTENSLAVE EQU 0A1H
DACPORT equ 301H

        .MODEL SMALL
        .DATA
                                        ;FILE VARIABLES
;*****************************************************************
FILE_NAME DB "c:\",25 DUP(?)
FILE_HANDLE DW 0                        ;FILE ERROR MESSAGES
;*****************************************************************
err_closefile DB "ERROR CLOSING THE FILE!","$"
err_openfile DB "ERROR OPENING THE FILE!","$"
err_readfile DB "ERROR READING THE FILE!","$"
;*****************************************************************
INPUT_CHAR DB "a"                       ;PROGRAM MESSAGES
;*****************************************************************
MSG_START_REQUEST DB "PRESS 'S' TO START PLAYING",13,10,"$"
MSG_PLAYING DB "PLAYING STARTED. PRESS 'S' AGAIN TO STOP OR 'P' TO
PAUSE",13,10,"$"
MSG_PAUSE DB 10,"PAUSED. PRESS 'P' AGAIN TO RESUME OR 'S' TO STOP",13,10,"$"
MSG_RESUME DB "RESUMED",13,10,"$"       ;DEBUG MESSAGES
```

```
;*********************************************************************
MSG_DEBUG1 DB "DEBUG: DATA REQUEST? ",10,"$"
MSG_DEBUG2 DB "DEBUG: CHECKING KEY ",10,"$"
MSG_DEBUG3 DB "DEBUG: SENDING: ","$"
MSG_DEBUG4 DB "DEBUG: END FO FILE ","$"
;*********************************************************************
        .CODE
MAIN:
        MOV AX,@DATA
        MOV DS,AX                   ;Taking the file name
;*********************************************************************
        XOR CX,CX
        MOV CL,ES:[80H]
        DEC CL
        MOV SI,2
        MOV DI,3

Input_File_Name:
        mov bl,es:[80h][si]
        mov FILE_NAME[DI],bl
        inc si
        inc di
loop Input_File_Name
        mov FILE_NAME[DI],0
        inc di
        mov FILE_NAME[DI],"$"           ;Open the file
;*********************************************************************
        mov ah,3dh
        mov al,0
        mov dx,offset FILE_NAME
        int 21h
jnc opensucceeded
        string err_openfile
        jmp endprog
opensucceeded:
        mov FILE_HANDLE,AX              ;Take User Command
;*********************************************************************
        string MSG_START_REQUEST
startcommand:
        mov ah,07h
        int 21h
        cmp al,"S"
jne startcommand                        ;Start reading from the file
;*********************************************************************
        string MSG_PLAYING
readloop:                       ;cHECK KEYBOARD
;-----------------------------------------
        mov ah,0bh
        int 21h
        cmp al,0
jz read_data
        mov ah,07h
```

```
        int 21h
        cmp al,"S"
jz close_file
        cmp al,"P"
jnz read_data
        string MSG_PAUSE
wait_command:
        mov ah,07h
        int 21h
        cmp al,"S"
jz close_file
        cmp al,"P"
jnz wait_command
        string MSG_RESUME
read_data:              ;CHECK DATA REQUEST FROM DAC
;-------------------------------------------------
;XXXXXXXXXXCHECK DATA REQUEST XXXXXXXXX
        int 255
        cmp al,0
jz end_of_readloop

;XXXXIF THERE IS REQUEST SEND DATA TO DAC XXXXXXXX

        mov al,INPUT_CHAR
        mov dx,DACPORT
        out dx,al
                        ;TAKE IN THE NEXT DATA FROM FILE
;-------------------------------------------------
        mov ah,3fh
        mov bx,FILE_HANDLE
        mov cx,1
        mov dx,offset INPUT_CHAR
        int 21h         ;SEND THE DATA TO DAC
;-------------------------------------------------
end_of_readloop:
jmp readloop

;Close the file
;*****************************************************************
close_file:
        mov ah,3eh
        mov bx,FILE_HANDLE
        int 21h
jnc closesucceeded
        string err_closefile
        jmp endprog
closesucceeded:                         ;Disable Interrupt on Master 8259
;*****************************************************************
      mov dx,INTENMASTER
      in al,dx
      or al,00000100b
      out dx,al                 ;Disable Interrupt 10 from Slave 8259
```

```
;*************************************************************************
        mov dx,INTENSLAVE
        in al, dx
        or al,00000100b
        out dx, al                      ;Terminate
;*************************************************************************


endprog:
          mov ah,4ch
          int 21h

END MAIN
```