Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧЕТ**

по лабораторной работе №8

по дисциплине «ЭВМ и переферийные устройства»

на тему «Таймер и звук»

Выполнили: студенты группы 22ВВП1

Захаров А. С.

Хоссейни Нежад С. А. С. М.

Приняли:

Никишин К. И.

Патунин Д. В.

Пенза 2025

**Цель работы**

Изучить принцип управления системным таймером и способы генерации звука.

**Задание**

1. Разработать и отладить программу генерации звукового сигнала. Генерация должна прекращаться с нажатием любой клавиши.
2. Модернизировать программу, обеспечивая изменение тона сигнала при каждом нажатии клавиши. Определить с помощью модернизированной программы значения коэффициента пересчета таймера, при которых обеспечивается высший и низший тон услышанного звукового сигнала.

**Ход работы**

**Задание 1:**

Листинг:

stseg segment para stack

dw 100h dup(?)

stseg ends

dseg segment para

curCount dw 0 ; текущий делитель для PIT (частота тона)

toneTable dw 5000, 4500, 4000, 3600, 3200, 2800, 2400, 2000, 1600, 1200 ; таблица значений делителя для цифр '0'..'9'

dseg ends

cseg segment para public 'CODE'

assume cs:cseg, ds:dseg, ss:stseg

main proc far

push ds

xor ax, ax

push ax

mov ax, dseg

mov ds, ax

key\_loop:

; Ждём нажатия клавиши (возвращает ASCII в AL)

call WaitKey

cmp al, 27 ; если ESC (код 27), выходим

je exit\_prog

; Проверяем, что введённый символ — цифра '0'..'9'

cmp al, '0' ; сравниваем с ASCII '0' (48)

jb key\_loop ; ниже '0' — не цифра, повторяем ожидание

cmp al, '9' ; сравниваем с ASCII '9' (57)

ja key\_loop ; выше '9' — не цифра, повторяем

; Преобразуем ASCII-код цифры в индекс 0–9

mov bl, al

sub bl, '0' ; BL = AL - '0'

xor bh, bh ; BH = 0, теперь BX = индекс цифры

; Получаем значение делителя из toneTable[index]

lea si, toneTable ; SI = адрес начала таблицы

mov ax, bx

shl ax, 1 ; умножаем индекс на 2 (слово = 2 байта)

add si, ax ; SI = адрес нужного элемента

mov ax, [si] ; AX = toneTable[index]

mov [curCount], ax ; сохраняем в curCount

; Запускаем воспроизведение тона

call PlayTone

stop\_wait:

; Ждём нажатия любой клавиши для остановки звука

call WaitKey

call StopTone

; Если нажали ESC — выходим, иначе — к новому вводу цифры

cmp al, 27

je exit\_prog

jmp key\_loop

exit\_prog:

pop ds

mov ah, 4Ch

int 21h

main endp

; WaitKey: ожидает нажатия клавиши через BIOS INT 16h, AH=00

; Выход: ASCII-код символа в AL

WaitKey proc near

mov ah, 00h

int 16h

ret

WaitKey endp

; PlayTone: настраивает канал 2 PIT для генерации квадратной волны (режим 3)

; Использует делитель из [curCount]

PlayTone proc near

mov ax, [curCount]

mov dx, ax ; сохраняем делитель в DX

; Отправляем команду на программирование канала 2: режим 3, LSB/MSB

mov al, 0B6h ; Команда для PIT: канал 2 (используется для звука), режим 3 (квадратная волна), LSB+MSB

out 43h, al ; Отправляем команду в управляющий регистр PIT (порт 43h)

; Передаём младший байт делителя

mov al, dl

out 42h, al

; Передаём старший байт делителя

mov al, dh

out 42h, al

; 42h — порт данных канала 2 таймера. PIT будет использовать загруженный делитель для генерации нужной частоты.

; Включаем динамик: устанавливаем биты 0 и 1 порта 61h

in al, 61h

or al, 03h

out 61h, al

;Порт 61h управляет динамиком на материнской плате.

;Бит 0 — управление динамиком (вкл/выкл)

;Бит 1 — разрешение сигнала с канала 2 таймера на динамик

ret

PlayTone endp

; StopTone: отключает динамик, очищая биты 0 и 1 порта 61h

StopTone proc near

in al, 61h

and al, 0FCh ; сбрасываем биты 0 и 1

out 61h, al

ret

StopTone endp

cseg ends

end main

Блок-схема программы:

**Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.**

**Задание 2:**

Листинг:

stseg segment para stack

dw 100h dup(?)

stseg ends

dseg segment para

curCount dw 3000 ; текущий делитель PIT (начальный тон)

stepSize dw 100 ; шаг изменения делителя при '+'/'-'

dseg ends

cseg segment para public 'CODE'

assume cs:cseg, ds:dseg, ss:stseg

main proc far

push ds

xor ax, ax

push ax

mov ax, dseg

mov ds, ax

; Запускаем тон

call PlayTone

adjust\_loop:

; Ожидаем нажатия клавиши: ASCII-код в AL

call WaitKey

cmp al, 27 ; ESC (код 27)?

je exit\_prog ; если да — выход

cmp al, '+' ; '+' — повысить тон?

je increase

cmp al, '-' ; '-' — понизить тон?

je decrease

jmp adjust\_loop ; остальные клавиши игнорируем

; Увеличение частоты (уменьшаем делитель)

increase:

mov ax, [curCount]

sub ax, [stepSize] ; уменьшаем делитель

cmp ax, 1 ; не ниже 1

jb adjust\_loop ; если ниже — пропускаем

mov [curCount], ax ; сохраняем новый делитель

call UpdateTone ; перенастраиваем PIT

jmp adjust\_loop

; Уменьшение частоты (увеличиваем делитель)

decrease:

mov ax, [curCount]

add ax, [stepSize] ; увеличиваем делитель

mov [curCount], ax ; сохраняем новый делитель

call UpdateTone ; перенастраиваем PIT

jmp adjust\_loop

; Завершение программы: выключаем тон, восстанавливаем DS, возвращаемся в DOS

exit\_prog:

call StopTone

pop ds

mov ah, 4Ch

int 21h

main endp

; WaitKey: BIOS int16h AH=00, ожидаем клавишу, возвращаем ASCII в AL

WaitKey proc near

mov ah, 00h

int 16h

ret

WaitKey endp

; UpdateTone: перенастройка канала 2 PIT без выключения динамика (режим 3)

UpdateTone proc near

mov ax, [curCount] ; загружаем текущий делитель

mov dx, ax ; сохраняем в DX для передачи байтов

mov al, 0B6h ; канал 2, режим 3, передача LSB/MSB

out 43h, al ; записываем в управляющий регистр PIT

mov al, dl ; младший байт делителя

out 42h, al ; порт данных канала 2

mov al, dh ; старший байт делителя

out 42h, al ; порт данных канала 2

ret

UpdateTone endp

; PlayTone: настройка PIT + включение динамика (порт 61h биты 0 и 1)

PlayTone proc near

mov ax, [curCount]

mov dx, ax ; делитель для передачи

mov al, 0B6h ; Команда для PIT: канал 2 (используется для звука), режим 3 (квадратная волна), LSB+MSB

out 43h, al ; Отправляем команду в управляющий регистр PIT (порт 43h)

mov al, dl ; младший байт

out 42h, al ; в порт 42h

mov al, dh ; старший байт

out 42h, al ; в порт 42h

in al, 61h ; читаем порт 61h

or al, 03h ; устанавливаем биты 0 (динамик) и 1 (PIT)

out 61h, al ; записываем обратно

ret

PlayTone endp

;Порт 61h управляет динамиком на материнской плате.

;Бит 0 — управление динамиком (вкл/выкл)

;Бит 1 — разрешение сигнала с канала 2 таймера на динамик

; StopTone: выключение динамика — сброс битов 0 и 1 порта 61h

StopTone proc near

in al, 61h

and al, 0FCh ; сбрасываем биты 0 и 1

out 61h, al

ret

StopTone endp

cseg ends

end main

Блок-схема программы:

**Изображение выглядит как снимок экрана, текст, Графика, графический дизайн

Контент, сгенерированный ИИ, может содержать ошибки.**

**Дополнительное задание:**

Листинг:

stseg segment para stack

dw 100h dup(?) ; резерв 256 слов для стека

stseg ends

dseg segment para

curDiv dw 0 ; текущий делитель для PIT (частота тона)

durCount dw 0 ; счётчик длительности текущей ноты (итерации delayStep)

pauseCount dw 60000 ; счётчик длительности паузы (~1 секунда)

; Таблица делителей для нот C, D, E, F, G, A, B

divTable dw 5000,4500,4000,3600,3200,2800,2400

; Последовательность мелодии: индексы в divTable или 0FFh для «длинной» паузы

melodyBytes db 6,3,6,3,6,5,5,0FFh

db 5,3,5,3,5,6,6,0FFh

db 6,3,6,3,6,5,5,0FFh

db 5,3,5,3,5,4,0FFh

; --- припев ---

db 4,5,5,5,5,0FFh

db 4,5,5,5,5,0FFh

db 5,6,5,6,5,6,0FFh

db 4,5,5,5,5,0FFh

db 4,5,5,5,5,0FFh

db 5,6,5,4,3,4,0FFh

totalNotes equ 69 ; общее число байт в melodyBytes

defaultDur dw 50000 ; стандартная длительность каждой ноты (~2 секунды)

dseg ends

cseg segment para public 'CODE'

assume cs:cseg, ds:dseg, ss:stseg

main proc far

push ds

xor ax, ax

push ax

mov ax, dseg

mov ds, ax

; Установка длительности нот и пауз

mov ax, [defaultDur]

mov [durCount], ax ; загрузить начальную длительность ноты

mov ax, [pauseCount]

mov [pauseCount], ax ; (может быть повторно инициализировано)

xor si, si ; SI = 0, индекс текущей ноты в melodyBytes

play\_sequence:

; Чтение следующего байта из melodyBytes

mov al, [melodyBytes + si]

cmp al, 0FFh ; проверка на маркер длинной паузы

je only\_pause ; если 0FFh — перейти к длинной паузе

; Вычисление делителя для текущей ноты

mov bl, al ; BL = индекс в divTable

xor bh, bh ; BH = 0

lea di, divTable ; DI = адрес начала таблицы делителей

mov ax, bx ; AX = индекс

shl ax, 1 ; умножаем на 2 (размер WORD)

add di, ax ; DI -> divTable[index]

mov ax, [di] ; AX = divTable[index]

mov [curDiv], ax ; сохраняем текущий делитель

; Проигрываем ноту заданной длительности

mov cx, [durCount] ; CX = количество итераций delayStep

call playToneDuration

; Корректная пауза между нотами

mov cx, [pauseCount]

call silence\_period

jmp next\_note

only\_pause:

; Выключаем динамик перед длинной паузой

call disable\_speaker

; Длинная пауза в два раза дольше обычной

mov cx, [pauseCount]

call silence\_period

mov cx, [pauseCount]

call silence\_period

next\_note:

inc si ; переходим к следующему байту melodyBytes

cmp si, totalNotes ; проверка конца мелодии

jb play\_sequence ; если ещё есть байты — повторяем

; Завершение программы: выход в DOS

pop ds

mov ah, 4Ch

int 21h

main endp

; enable\_speaker: включить динамик с частотой curDiv

enable\_speaker proc near

mov ax, [curDiv] ; загружаем делитель

mov dx, ax ; DX для байтов делителя

mov al, 0B6h ; Команда для PIT: канал 2 (используется для звука), режим 3 (квадратная волна), LSB+MSB

out 43h, al ; управляющий регистр PIT

mov al, dl ; младший байт делителя

out 42h, al ; порт данных канала 2

mov al, dh ; старший байт делителя

out 42h, al ; порт данных канала 2

in al, 61h ; читаем порт управления динамиком

or al, 03h ; включаем биты 0 (динамик) и 1 (PIT)

out 61h, al

ret

enable\_speaker endp

;Порт 61h управляет динамиком на материнской плате.

;Бит 0 — управление динамиком (вкл/выкл)

;Бит 1 — разрешение сигнала с канала 2 таймера на динамик

; disable\_speaker: отключить динамик (сброс битов 0 и 1 порта 61h)

disable\_speaker proc near

in al, 61h

and al, 0FCh

out 61h, al

ret

disable\_speaker endp

; delayStep: одна итерация задержки (NOP — можно расширить для точного тайминга)

delayStep proc near

nop

ret

delayStep endp

; playToneDuration: воспроизвести тон в течение CX итераций delayStep

playToneDuration proc near

call enable\_speaker

tone\_loop:

call delayStep

loop tone\_loop ; повторять until CX=0

call disable\_speaker

ret

playToneDuration endp

; silence\_period: пауза без звука, CX итераций delayStep

silence\_period proc near

silence\_loop:

call delayStep

loop silence\_loop

ret

silence\_period endp

cseg ends

end main

Блок-схема программы:

**Изображение выглядит как текст, снимок экрана, Шрифт, черно-белый

Контент, сгенерированный ИИ, может содержать ошибки.**

**Вывод**

Изучили принцип управления системным таймером и способы генерации звука.