Государственное образовательное учреждение
высшего профессионального образования
"САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ им. академика С.П. КОРОЛЕВА"

ГЕНЕРАЦИЯ ЗВУКА НА ПЭВМ

Методические указания к лабораторному практикуму
по курсу «Системы реального времени»

Составитель: К.Е. Климентьев

Самара 2004

Составитель: К.Е. Климентьев

УДК 681.3+618.3.07

Генерация звука на ПЭВМ: Метод. Указания / Самар. Гос. Аэрокосм. ун-т; Сост. К.Е. Климентьев. Самара, 2004. 23 с.

Предназначены для студентов, изучающих в рамках специальности 230102 курс «Системы реального времени». Также могут быть использованы при изучении курсов смежной тематики, в курсовом и дипломном проектировании.

Содержат справочный материал, необходимый для программирования звука на ПЭВМ: описания устройств, команд управления, интерфейсов программирования.

Выполнены на кафедре информационных систем и технологий.

Печатается по решению редакционно-издательского совета СГАУ.

Рецензент ....

Содержание

Введение

Звук - колебания упругой среды (газа, жидкости, твердого тела), воспринимаемые человеческим ухом. Человек способен воспринимать звуки с колебаниями от 20 до 20000 Гц. Информация содержится как в частоте звуковых колебаний, так и в их амплитуде и фазе.

Музыкальные звуки представляют собой гармонические колебания определенной частоты. Условно диапазон музыкальных звуков разделяют на октавы, каждая из которых состоит из 7 основных нот («до», «ре», «ми» и т.п.) и 5 располагающихся между ними полутонов («до-диез», «ре-диез» и т.п.) - т.е. всего из 12 звуков (см. рис. 1). Внутри октавы периоды колебаний соседних звуков отличаются друг от друга в $\sqrt[12]{2}$ раз. Периоды и частоты колебаний одноименных звуков, находящихся в соседних октавах, отличаются в 2 раза (например, «до» первой октавы соответствует частота 131 Гц, а «до» второй октавы - 262 Гц).

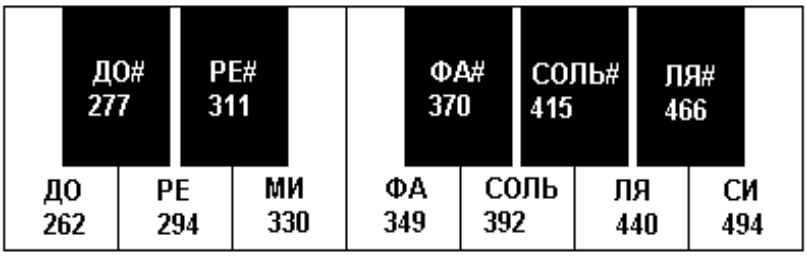


Рис. 1. Ноты и соответствующие им частоты второй октавы

1. Использование встроенного динамика ПЭВМ

Встроенный динамик ПЭВМ представляет собой громкоговоритель, на вход которого поступает сигнал OUT от канала системного таймера с номером 2. Поскольку сигнал OUT может иметь только два значения - 0 и 1, то воспроизводимый динамиком звук возбуждается колебаниями прямоугольной формы и постоянной амплитуды. Частота звуковых колебаний соответствует частоте изменения сигнала OUT и может быть изменена программно в интервале от 18.2 Гц до 1.19 МГц. Наличие или отсутствие звучания динамика может управляться битами порта 61h:

- бит 1 включает/выключает прохождение управляющего сигнала OUT на динамик;
- бит 0 включает/выключает работу таймерного счетчика.

1.1. Метод, основанный на программировании счетчика

«Стандартный» метод генерации музыкальных звуков основан на управлении частотой изменения сигнала OUT.

Шаг 1. Поместить в справочную константу 2-го канала таймера значение, соответствующее требуемой частоте звучания. Пример: для «до» первой октавы частота изменения должна составлять $f_{ДО}$=131 герц, что соответствует периоду $T_{ДО}$=1/f=0.007633 с. Зная период тактовых импульсов, поступающих на таймерный счетчик $T_{СЧ}$=0.00000083 с, можно определить количество этих импульсов в одном периоде колебаний: $N=T_{ДО}/T_{СЧ}$=9196, что и представляет собой значение коэффициента пересчета, которое нужно загрузить в счетчик таймера.

Шаг 2. Разрешить работу счетчика 2-го таймерного канала и включить динамик.

Шаг 3. Выполнить временную задержку на интервал времени, соответствующий длительности звучания ноты.

Шаг 4. Выключить счетчик и динамик.

Пример программирования встроенного динамика приведен в Приложении Д.

1.2. Метод, основанный на коммутации сигнала «OUT»

Метод основан на включении/выключении прохождения сигнала OUT на динамик.

Шаг 1. Выключить счетчик 2-го канала таймера.

Шаг 2. Включить динамик на интервал времени, соответствующий длительности полупериода колебаний. Пример: для «до» первой октавы это 0.003817 с.

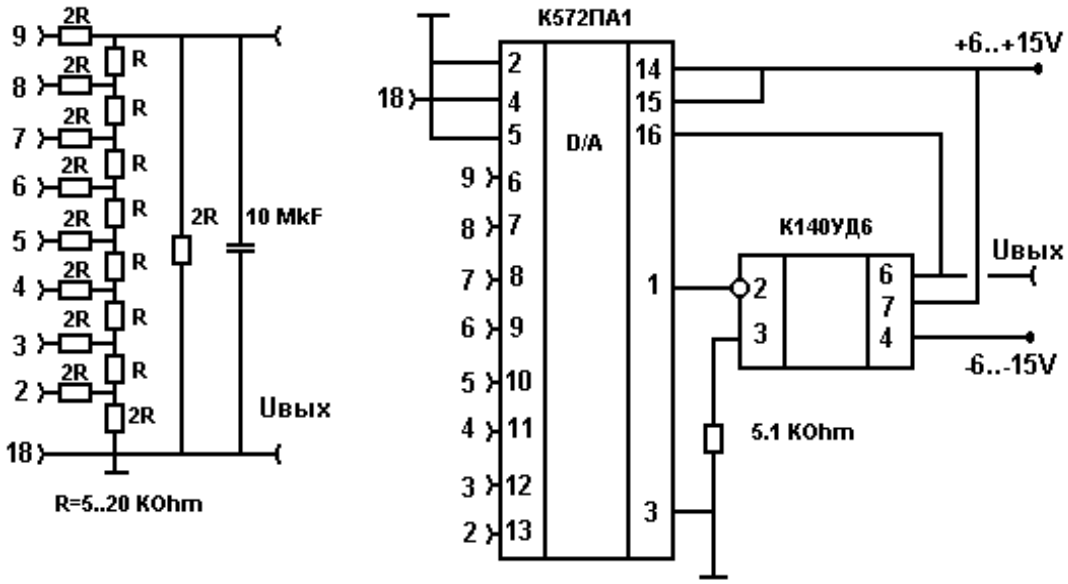
Шаг 3. Выключить динамик на интервал времени, соответствующий длительности полупериода колебаний.

Шаг 4. Повторить Шаги 1,2,3 необходимое количество раз, чтобы обеспечить желаемую длительность звучания ноты. Пример: для звучания ноты «до» первой октавы в течение T=0.5 с, необходимо выполнить цикл $N=T/T_{ДО}$ =0.5/0.007633=66 раз.

Шаг 5. Выключить динамик.

2. Использование устройства «Covox»

Устройство «Covox» представляет собой интерфейс между параллельным («принтерным») портом ПЭВМ и динамиком. Параллельный порт ПЭВМ имеет 8 линий выходных данных, информация на которые выдается в виде напряжений с уровнями в стандарте TTL. Назначение устройства «Covox» - пропорциональное преобразование двоичного числа (в интервале 0..255), сформированного на выходных линиях LPT-порта, в напряжение (имеющее 256 возможных градаций), подаваемое на вход динамика. Возможны два варианта устройства «Covox» (см. Рис. 2).



а) на резисторах б) на микросхеме ЦАП

Рис. 2. Схемы устройства «Covox»

Устройство «Covox» позволяет непосредственно формировать сигнал напряжения необходимой формы и амплитуды, в том числе и гармонические колебания.

Пример. Пусть необходимо сформировать синусоиду с частотой f=131 Гц, что соответствует ноте «до» первой октавы.

Шаг 1. Вычислить период синусоидальных колебаний $T_{ДО}=1/f=0.007633$ с.

Шаг 2. Выбрать количество отсчетов, моделирующих один период синусоиды, например, N=16. Соответственно, период дискретизации составит $T_д= T_{ДО}/N=0.000477$с (см. рис. 3).

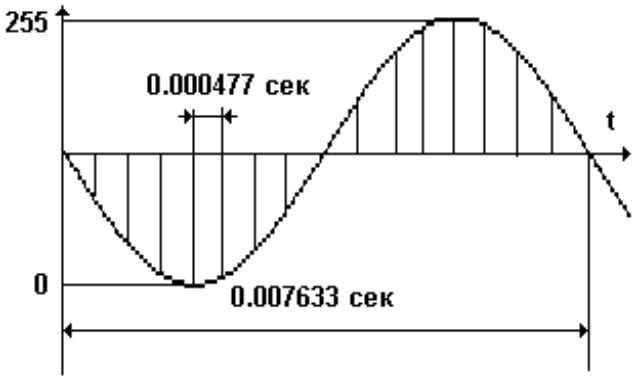


Рис. 3. Моделирование синусоиды

Шаг 3. Положить значение абсолютного времени t=0.

Шаг 4. Рассчитать очередное значение амплитуды сигнала A=sin(t), сформировать из него целое значение K:=[(A+1)*127.5] и послать в параллельный порт.

Шаг 5. Выполнить временную задержку на $T_д$=0.000477 с. Увеличить текущее значение абсолютного времени t := t + (2π/N).

Шаг 6. Повторить Шаги 4 и 5 необходимое количество раз (N=16) для завершения формирования периода синусоиды.

Шаг 7. Повторить шаги 3, 4, 5 и 6 несколько раз для обеспечения желаемой длительности звучания ноты. Например, для длительности 0.5с необходимо выполнить цикл из 66 итераций (см. п. 1.2).

3. Использование звуковых карт

Современные звуковые карты представляют собой сложные многофункциональные устройства, содержащие высокоточные (разрядность 16-24 бита) быстродействующие (время преобразования 5-25 мкс) АЦП и ЦАП (см. рис. 4). На этом рисунке: «LINE IN» - линейный вход, «MIC» - вход с микрофона, «LINE OUT» - линейный выход, «SPKR» - выход на наушники или внешние динамики, «Ф» - аналоговые фильтры, «У» - усилители аналоговых сигналов, «DSP» - процессор цифровых сигналов. На DSP возложено большое количество задач по управлению процессами аналогово-цифрового и цифро-аналогового преобразования, по синтезу и обработке цифровых сигналов, по обмену данными с ПЭВМ и пр.

Примечание. Современные устройства на основе интерфейса AC97 содержат только встроенные ЦАП и АЦП, а функции DSP реализуются программно драйверами операционной системы.

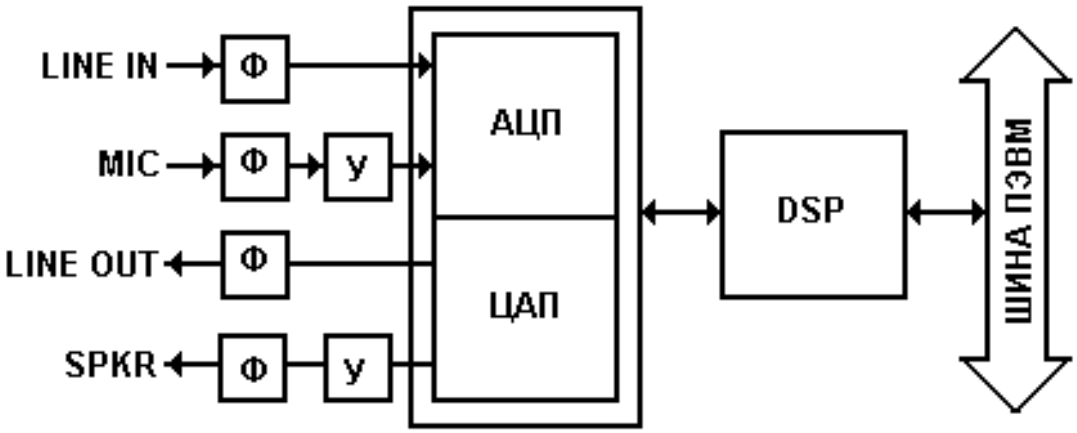


Рис. 4. Функциональная схема одноканальной звуковой платы

Исторически первые звуковые карты (модель SoundBlaster фирмы Creative Technology, первая половина 90-х годов XX века) были одноканальными. Благодаря мультиплексированию цепей преобразования аналоговой информации количество каналов ввода-вывода звука в современных устройствах может достигать 2÷6.

Звуковые карты используют следующие системные ресурсы ПЭВМ:

- 16-портов ввода-вывода, начиная с базового адреса (обычно BASE=220h для 1-го канала и 222h - для второго);

- 2 (в отдельных случаях 4) порта ввода-вывода, начиная с адреса 388h, предназначенные для программирования «Adlib»-режимов;
- одно аппаратное прерывание (обычно это Irq5/Int 0Dh или Irq10/Int 72h);
- один канал контроллера ПДП (обычно это канал 1).

Конкретные значения номеров портов, прерываний и каналов ПДП устанавливаются перемычками на звуковой плате или программно. Для того, чтобы прикладное программное обеспечение получило простой доступ к этим значениям, рекомендуется в конфигурационном файле (для MS-DOS это AUTOEXEC.BAT) проинициализировать переменную окружения BLASTER с параметрами: A - базовый адрес области портов ввода-вывода, I - номер аппаратного прерывания, D - номер канала контроллера PDP, например:

SET BLASTER=A220 I5 D1.

Программное управление режимами работы DSP осуществляется через порты ввода-вывода (см. табл. 1) при помощи специальных команд (см. табл. 2).

Таблица 1

Некоторые порты ввода-вывода звуковой карты

Порт	Назначение
BASE+6	Регистр управления DSP
BASE+0Ah	Буферный регистр для чтения данных
BASE+0Ch	1. Буферный регистр для записи данных и команд 2. Регистр состояния буфера записи (если бит 7 сброшен, то разрешена запись в буферный регистр)
BASE+0Eh	Регистр состояния буфера чтения (если бит 7 установлен, то разрешено чтение из буферного регистра)

Таблица 2

Некоторые команды управления DSP

Команда	Описание
0Bxh	Установка параметров ввода/вывода через ПДП (16 битов)
0Cxh	Установка параметров ввода/вывода через ПДП (8 битов)
10h	Непосредственная передача байта на ЦАП
20h	Непосредственное чтение байта с АЦП
14h	Вывод звука в режиме ПДП (8 битов)
1Ch	Вывод звука в режиме ПДП с автоинициализацией (8 битов)
24h	Ввод звука в режиме ПДП (8 битов)
40h	Установка частоты автозапуска АЦП/ЦАП через код
41h	Установка частоты автозапуска ЦАП непосредственно
42h	Установка частоты автозапуска АЦП непосредственно
48h	Установка размера передаваемых данных
0D1h	Разрешение ввода-вывода звука

0D3h	Запрет ввода-вывода звука
0D9h	Завершение работы через ПДП (16 битов)
0Dah	Завершение работы через ПДП (8 битов)
0E1h	Запрос версии DSP (1 – SB, 2 - SB 2.0, 3 - SB PRO, 4 - SB 16)

Инициализация DSP состоит из следующих действий:

- послать значение 1 в порт BASE+6;
- выполнить временную задержку на интервал не менее 3 мкс;
- послать значение 0 в порт BASE+6.

Проверка исправности. После окончания инициализации (иногда с задержкой на несколько десятков мкс) правильно сконфигурированная звуковая карта при чтении из DSP возвращает значение 0AAh.

Чтение из DSP заключается в том, чтобы:

- в цикле ожидать готовности DSP (установки 7-го бита в порту BASE + 0Eh);
- прочитать байт из порта BASE+0Ah.

Аналогично выглядит *запись в DSP*:

- в цикле ожидать готовности DSP (сброса 7-го бита в порту BASE +0Ch);
- записать байт в порт BASE+0Ch.

3.1. Режим непосредственного ввода/вывода

DSP поддерживает ряд сохранившихся еще с младших моделей Creative Soundblaster режимов работы звуковой карты, основанных на непосредственной работе с ЦАП и АЦП. Эти режимы предусматривают принудительный программный запуск одиночного цифро-аналогового или аналогово-цифрового преобразования. Разрядность данных в этих режимах составляет 8 битов.

Разрешение ввода-вывода заключается в посылке в DSP команды 0D1h, а *запрещение ввода-вывода* - в посылке команды 0D3h.

Чтение данных из АЦП звуковой карты складывается из следующих действий:

- послать в DSP команду с кодом 20h;
- прочитать байт данных из DSP.

В противоположность этому, *посылка данных* на ЦАП звуковой карты заключается в том, чтобы:

- послать в DSP команду с кодом 10h;
- послать в DSP байт данных.

Пример программирования звуковой карты в режиме прямого доступа приведен в Приложении Е.

3.2. Режим доступа через ПДП

DSP звуковых карт поддерживает ряд режимов, обеспечивающих обмен звуковой информацией с оперативной памятью ПЭВМ без участия процессора (см. Приложение В). При этом звуковая карта самостоятельно выполняет необходимое количество цифро-аналоговых и аналогово-цифровых преобразований с программно заданной частотой. Среди них есть режимы, использующие передачу 16-битовых данных, 8-битовых данных и данных, подвергнутых сжатию по технологии ADPCM (последняя группа режимов здесь не рассматривается).

Шаг 1. Завести в оперативной памяти массив, заполнить его данными. Массив не должен превышать по размеру 64К, и не должен пересекать границу физического сегмента памяти.

Шаг 2. Во всех режимах ввода-вывода через контроллер ПДП звуковая карта по окончании приема/передачи блока данных инициирует аппаратное прерывание (с номером, зависящем от конфигурации карты, часто это Int 0Dh). Поэтому целесообразно установить собственный обработчик этого прерывания, который предназначен для повторной инициализации DSP, обслуживания массива данных (обновления при передаче и сохранения при приеме) и, возможно, для организации завершения программы.

Шаг 3. Установить период автоматического перезапуска ЦАП:

- послать в DSP команду 40h;
- послать в DSP код частоты f, вычисляемый по формуле 256-1000000/f (например, для частоты 10 кГц это 256-100=156).

Альтернативный способ:

- послать в DSP команду 41;
- послать в DSP последовательно младший и старший байты частоты f дискретизации (допустимы значения в интервале 5..44 КГц).

Шаг 4. Установить режим работы звуковой карты. Для организации *простого 8-битового* ввода/вывода:

- послать в DSP код команды 14h (для вывода сигнала) или 24h (для ввода);
- послать в DSP последовательно младший и старший байты длины блока данных минус 1.

Для организации режима *8-битового вывода с автоинициализацией* (т.е. режима непрерывного звучания):

- послать в DSP код команды 1Ch;
- задать размер блока данных посылкой кода команды 48h, а также старшего и младшего байтов длины блока данных минус 1.

Универсальный способ установки режима работы доступен в звуковых картах класса SB 16, он основан на использовании команд Cxh (8-битовый ввод/вывод) и Bxh (16-битовый ввод-вывод):

- послать в DSP команду C4h/B4h для вывода сигнала или CCh/BCh для ввода (сброс бита 2 включает автоинициализацию);
- послать в DSP значение 20h (для стереозвука) или 0 (для моно);
- послать в DSP последовательно младший и старший байт числа байтов или слов в массиве данных минус 1.

Шаг 5. Проинициализировать контроллер ПДП, указав в параметрах инициализации адрес массива и его длину (см. Приложение В). После выполнения этого шага начнется обмен данными между оперативной памятью и звуковой картой.

3.3. Режим волнового синтеза (режим Adlib)

Устройство «Adlib» - это программно-управляемый синтезатор гармонических сигналов произвольной формы, разработанный в 80-х годах XX века и предназначенный для подключения к ПЭВМ. В настоящее время большинство звуковых карт самостоятельно моделируют основные функции этого устройства (это режим «FM-синтеза»).

Контроллер устройства «Adlib» содержит 244 внутренних регистра, пронумерованных от 1 до 244. Доступ к этим регистрам осуществляется через два порта ввода-вывода:

- 388h - индексный порт, в него надо послать номер интересующего регистра;
- 389h - порт данных, через него производится чтение/запись значений интересующего регистра.

Кроме того, порт 388h используется в качестве регистра состояния таймеров.

Доступ ко внутренним регистрам устройства «Adlib» осуществляется в результате следующей последовательности действий:

- послать в порт 388h номер интересующего регистра;
- выполнить временную задержку не менее 3.3 мкс.;
- записать необходимое значение в порт 389h;
- выполнить временную задержку не менее 23 мкс.

Примечание. В фирменной документации допускается для выполнения упомянутых выше временных задержек циклически прочитать порт 7 и 46 раз соответственно.

Сброс устройства «Adlib» заключается в загрузке значения 0 во все 244 внутренних регистра.

Таймеры. Устройство «Adlib» содержит два 8-разрядных инкрементирующих таймера. Счетчик первого из них проецируется на регистр с номером 02 и инкрементируется каждые 80 мкс, счетчик другого - на регистр 03 и имеет период инкремента 320 мкс. При переполнении любого из счетчиков в порту 388h устанавливается старший бит, а также бит номер 6, если

переполнился первый таймер, или бит номер 5, если второй. Для управления таймерами служит регистр с номером 04:

- бит 7 - сброс таймерных битов в порту 388h;
- бит 6 - сброс таймера 1;
- бит 5 - сброс таймера 2;
- бит 2 - запуск таймера 2;
- бит 1 - запуск таймера 1.

Проверка исправности устройства «Adlib» основана на тестировании работы внутренних таймеров, например, первого, и выполняется следующим образом.

Шаг 1. Сбросить таймеры, последовательно записав в регистр 04 значения 60h и 80h. Проверить, что в порту 388h в результате этого оказываются сброшены биты 5, 6 и 7.

Шаг 2. Загрузить в счетчик первого таймера (в регистр с номером 02) максимально возможное значение 0FFh.

Шаг 3. Запустить первый таймер (записать значение 21h в регистр 04).

Шаг 4. Выполнить временную задержку в 80 мкс.

Шаг 5. Убедиться, что произошло переполнение счетчика таймера, и теперь в порту 388h установлены биты 6 и 7.

Модель звукового сигнала. Предполагается, что звуковой сигнал представляет собой модуляцию высокочастотной несущей частоты при помощи низкочастотных звуковых колебаний с огибающей сложной формы, на которой выделяются следующие участки (см. рис. 5):

- атака - интервал нарастания амплитуды;
- остановка - кратковременный интервал с максимальной амплитудой;
- спад - интервал перехода в установившееся состояние;
- удержание - интервал установившегося состояния;
- затухание - интервал уменьшения амплитуды до минимума.

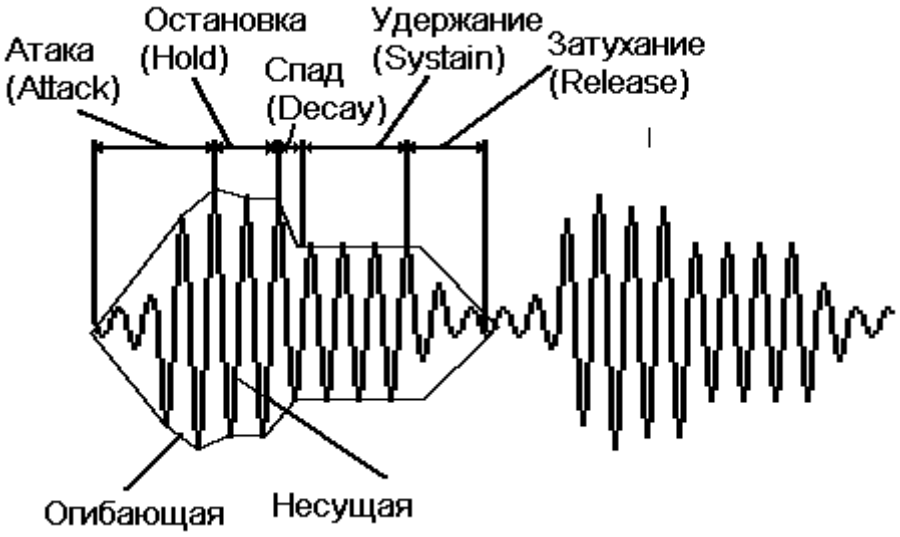


Рис 5. Модель звукового сигнала

Инструменты. Устройство «Adlib» может моделировать одновременное звучание до 9 различных инструментов. Управляющие регистры, отвечающие за каждый параметр звучания, оформлены в виде нескольких групп из последовательно расположенных адресов.

Группы первого типа содержат по 9 регистров (см. табл 3). Внутри группы каждый регистр отвечает за отдельный инструмент.

Таблица 3

Регистры групп первого типа

Регистры	Описание
A0h-A8h	Биты 0-8 : младшая часть кода ноты (см. табл 4).
B0h-B8h	Биты 0-1 : старшая часть кода ноты (см. табл 4); Биты 2-4 : номер октавы от 0 до 7; Бит 5 : включение/выключение звучания ноты.
C0h-C8h	Биты 1-3: 000 - обратной связи нет, иначе - текущее значение сигнала зависит от "истории". Бит 0: 0 - вторая цепь является несущей для огибающей, задаваемой первой цепью, 1 - обе цепи независимы.

Таблица 4

Кодирование нот в 4 октаве

До#	Ре	Ре#	Ми	Фа	Фа#	Соль	Сол#	Ля	Ля#	Си	До
16B	181	198	1B0	1CA	1E5	202	220	241	263	281	2AE

Группы второго типа содержат по 22 регистра каждая (см. табл.6). Предполагается, что для описания звучания каждого инструмента используются два различных гармонических сигнала (две *цепи*). Внутри группы регистры распределяются в соответствии с табл. 5.

Таблица 5

Распределение регистров внутри группы

Инструмент	00	01	02	03	04	05	06	07	08
Цепь 1	+00	+01	+02	+08	+09	+0A	+10	+11	+12
Цепь 2	+03	+04	+05	+0B	+0C	+0D	+13	+14	+15

Например, для настройки инструмента №5 используются следующие регистры:

- 29h - цепь 1, тремоло;
- 2Ch - цепь 2, тремоло;
- 49h - цепь 1, уровень;
- 4Ch - цепь 2, уровень;
- 69h - цепь 1, атака;
- 6Ch - цепь 2, атака;
- 89h - цепь 1, затухание;
- 8Ch - цепь 2, затухание;
- A4h - частота ноты (8 младших битов);

- B4h - включить ноту / октава / частота ноты (2 старших бита);
- C4h - величина обратной связи / режим объединения цепей;
- E9h - цепь 1, форма волны;
- ECh - цепь 2, форма волны.

Таблица 6

Регистры второй группы

Регистры	Описание
20h-35h	Искажения сигнала: - бит 7 : тремоло вкл/выкл.; - бит 6 : вибрато вкл/выкл.; - бит 5 : управление интервалом удержания; - бит 4 : управление увеличением длительности сигнала с ростом частоты; - биты 0-3: номер гармоники (частоты, кратной основной частоте сигнала). Рекомендуемое значение: 1.
40h-55h	Громкость: - биты 0-5 управляют затуханием относительно базового уровня: бит 0: 0.75 Дб; бит 1: 1.5 Дб; бит 2: 3 Дб; бит 3: 6 Дб; бит 4: 12 Дб; бит 5: 24 Дб, все 0 дают максимальную громкость; - биты 6-7 управляют неравномерностью АЧХ: 00 - никак, 01 - 3 Дб/Гц; 10 - 1.5 Дб/Гц; 11 - 6 Дб.
60h-75h	Параметры атаки и спада: - биты 0-3 - скорость спада; - биты 4-7 - скорость атаки.
80h-95h	Параметры удержания и затухания: - биты 0-3 - скорость затухания; - биты 4-7 - уровень удержания.
E0h-F5h	Биты 0-1 : выбор формы несущей (см. рис. 6).

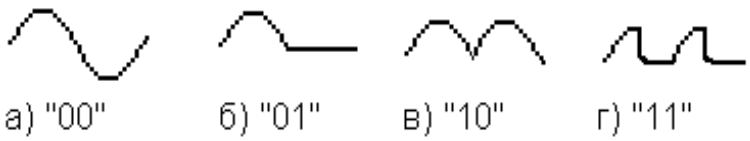


Рис 6. Формы несущей частоты

Ударные инструменты. Описанных выше инструментов недостаточно для создания перкуссионных звуков, поэтому для них используется специальный регистр 0BDh:

- бит 0 - тарелки;
- бит 1 - цимбалы;
- бит 2 - тамтам;
- бит 3 - малый барабан;
- бит 4 - большой барабан;
- бит 5 - режим ударных (если 0, то биты 0-4 игнорируются);

- бит 6 - вибрато;
- бит 7 - тремоло.

Пример программирования устройства AdLib приведен в Приложении Д.

3.4. Программирование звуковых карт в Windows

В операционных системах семейства Windows прямой доступ прикладных программ к устройствам через порты ввода-вывода запрещен.

Примечание 1. Прямое программирование звуковых карт и динамика через порты ввода-вывода в Windows все же возможно с использованием драйверов типа OPENPORT, TWICHW32 и пр., а так же в специализированных инструментальных средах типа LabView.

Примечание 2. Существуют специализированные виртуальные машины (например, DosBox), которые очень качественно моделируют работу процессора в реальном режиме, доступ к портам ввода-вывода, обработку прерываний, особенности устаревших видеоадаптеров, звуковых карт и т.п. «Музыкальные» программы, предназначенные для работы в MS-DOS, можно запускать именно в среде этих виртуальных машин.

Тем не менее, существуют несколько программных интерфейсов, позволяющих генерировать звук с использованием звуковых карт. Рассмотрим простейший интерфейс «MIDI», сконцентрированный в библиотеке «WINMM.DLL».

Способ 1. Воспроизведение WAV-файла. Зная его внутренний формат (см. Приложение Г), можно рассчитать (см. раздел «Использование устройства Covox»), сгенерировать и записать в файл желаемую последовательность амплитуд сигнала, которая будет воспроизведена ЦАП звуковой карты. Проигрывание файла осуществляется при помощи API-функции

- BOOL sndPlaySound(char *filename, unsigned int flags).

Первый параметр – строка имени файла, содержащего звук; либо NULL, что прерывает любое звучание; либо указатель на область памяти. Если файл со звуком не найден, звучит «стандартный» сигнал.

Второй параметр может комбинироваться из следующих битовых флагов: SND_ASYNC – асинхронное выполнение (т.е. звучание параллельно работе программы); SND_SYNC – синхронное выполнение (т.е. с задержкой до завершения звука); SND_LOOP – зацикленное звучание; SND_MEMORY – указывает, что источником звука является область памяти.

Способ 2. Волновой синтез. Генерация звука в этом режиме базируется на работе следующих API-функций:

- midiOutOpen() –открывает звуковой канал;
- midiOutShortMsg() – посылает устройству команды;
- midiOutClose() – закрывает звуковой канал.

Совместная работа этих функций проиллюстрирована в Приложении Д, пояснения требует только второй параметр функции midiOutShortMsg(), это 32-битовое целое число вида 0xTTXXYYZZ. Отдельные байты его несут следующую нагрузку.

TT – это номер звукового канала, по умолчанию 00.

Если ZZ=0xC0, то производится выбор одного из 128 заранее предопределенных музыкальных инструментов. В этом случае YY – номер инструмента, вот некоторые: 0,1,2,3,4 – различные варианты фортепиано; 7 – клавесин; 8 – челеста; 0x0D – ксилофон; 0x16 – гармонь; 0x18, 0x19, 0x1A – различные гитары; 0x40, 0x41 – саксофоны; 0x39 - тромбон, 0x7B - птица, 0x7D - звонок, 0x7F – ружейный выстрел. Например, значение 0x00000DC0 выбирает ксилофон.

Если ZZ=0x90, то запускается звучание выбранным инструментом ноты, определяемой байтом YY. Фактически, YY - это номер клавиши на длинной, многооктавной клавиатуре. Например, для одной из октав 0x3C это «до», 0x3E это «до#», 0x40 это «ре» и так далее. Байт AA при этом определяет длительность звука, где 0x7F – максимальная. Например, значение 0x007F4190 запускает максимально долгое звучание ноты «ре#».

Если ZZ=0x80, то звучание прерывается. Остальные байты описаны выше.

Приложение А. Устройство и программирование системного таймера

Системный таймер современных ПЭВМ - это устройство, позволяющее вычислительной системе решать задачи *реального времени*, а именно:

- отслеживать моменты наступления программно-аппаратных событий;
- самостоятельно генерировать такие события в требуемые моменты времени.

Функционально системный таймер может быть представлен в виде совокупности трех “каналов”, работающих независимо друг от друга (см. рисунок А.1). Работа каждого из каналов синхронизирована с тактовыми импульсами, поступающими с кварцевого генератора с частотой 1,193180 МГц.

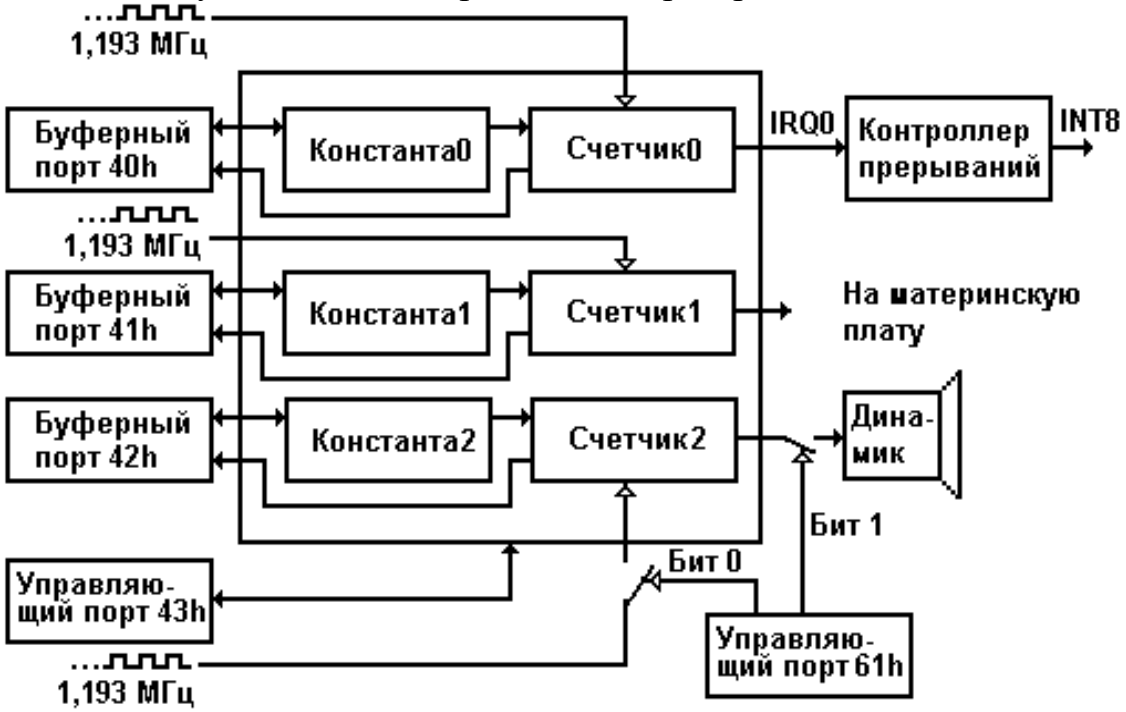


Рис. А.1. Функциональная схема системного таймера

Каждый из каналов может работать в одном из шести режимов. По умолчанию для первого и последнего каналов установлен режим №3. В этом режиме каналы работают следующим образом (см. рисунок А.2):

- в счетчик таймера загружается некоторая константа пересчета в диапазоне 1..65535 (по умолчанию она имеет максимально возможное значение);
- по фронту каждого приходящего тактового импульса из счетчика вычитается значение 2;
- при достижении счетчиком значения 0 на выходе у канала таймера инвертируется значение сигнала «OUT», кроме того в счетчик вновь загружается исходная константа и работа счетчика продолжается;
- по фронту сигнала «OUT» (т.е. при каждом втором обнулении счетчика) генерируется некоторое “событие”.

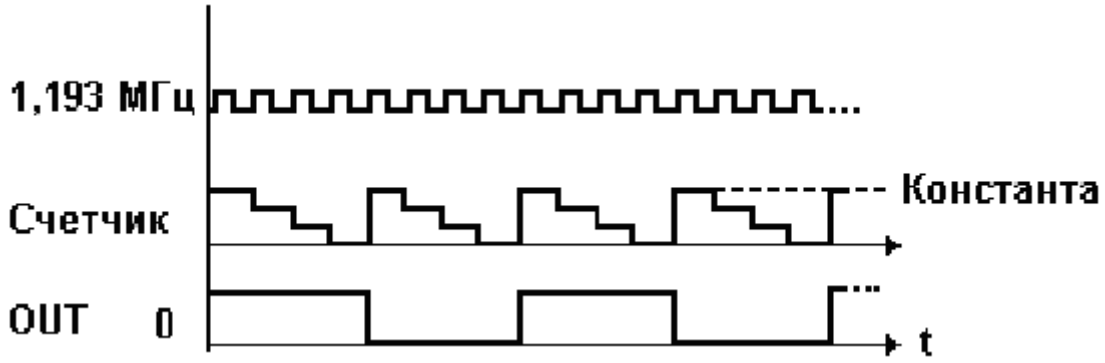


Рис. А.2. Временная диаграмма работы счетчика

Сигнал «OUT» канала 0 подается на контроллер прерываний по входу Irq0 и, соответственно, по фронту каждой прямоугольной «волны» в системе возникают прерывания с номером 8.

Канал 1 синхронизирует системные процессы на материнской плате ПЭВМ.

Сигнал «OUT» канала 2 представляют собой прямоугольные «волны», подающиеся на вход встроенного компьютерного динамика. Управление работой канала и звучанием динамика осуществляется через порт 61h: бит 0 маскирует поступление в счетчик канала тактовой частоты, а бит 1 - маскирует передачу выходного сигнала канала на динамик.

Все три канала являются программируемыми: возможно переключение режимов работы канала, изменение константы пересчета, считывание и загрузка значений счетчиков и пр. Управление таймером осуществляется через порт 43h (см. таблицу А.1).

Таблица А.1

Управляющий регистр системного таймера

Биты	Назначение
0	Тип счета: 0 - двоичный , 1- двоично-десятичный
1..3	Режим работы канала: 011 - режим №3.

4..5	Команда: 00 - зафиксировать значение счетчика в буферном регистре; 01 - подготовить чтение/загрузку старшего байта счетчика; 10 - подготовить чтение/загрузку младшего байта счетчика; 11 - подготовить последовательное чтение/загрузку младшего и старшего байтов счетчика.
6..7	Номер канала: 00 - нулевой, 01-первый, 10-второй.

Этот же порт служит для чтения состояния канала (см. таблицу А.2). Режим использования порта 43h определяется значением двух его старших битов.

Таблица А.2

Регистр управления чтением состояния счетчика

Биты	Назначение
0	0
1	1 - выбор нулевого канала, 0 - нет
2	1 - выбор первого канала, 0 - нет
3	1 - выбор второго канала, 0 - нет
4	0 - подготовить чтения слова состояния каналов, 1 - нет
5	0 - подготовить чтение содержимого счетчиков, 1 - нет
6..7	11 - признак команды чтения состояния счетчика

Обмен данными с каналами счетчика выполняется через порты 40h (нулевой канал), 41h (первый канал), 42h (второй канал).

Инициализация канала новой константой пересчета заключается в следующем:

- записать в порт 43h управляющий байт со значением kk110110 (kk - номер канала);
- последовательно записать в буферный регистр данных младший и старший байты константы пересчета.

Чтение счетчика можно выполнить так:

- зафиксировать в буферном регистре содержимое счетчика, послав в порт 43h значение kk000110 (kk - номер канала);
- подготовить чтение счетчика, послав в порт 43h управляющий байт со значением 1101nnn0 (nnn - биты разрешения каналов);
- последовательно прочитать из соответствующего буферного порта сначала младший, а затем старший байты значения счетчика.

Чтение состояния канала производится следующим образом:

- подготовить чтение состояния, послав в порт 43h управляющий байт со значением 1110nnn0 (nnn - биты разрешения каналов);
- прочитать байт состояния из соответствующего буферного регистра (бит 7 - значение сигнала OUT, биты 1..3 - текущий режим работы).

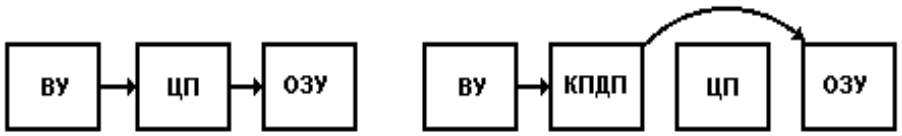
Приложение Б. Устройство и программирование контроллера ПДП

Прямой доступ к памяти (ПДП, или DMA - Direct Memory Access) - это способ организации обмена данными между внешним устройством (например, жестким диском) и оперативной памятью, осуществляемый без участия центрального процессора (см. рис. В.1). Контроллер ПДП современных ПЭВМ обслуживает 7 каналов обмена данными (см. табл. В.1).

Таблица В.1

Каналы контроллера ПДП

Канал	Разрядность	Описание
0	8	Зарезервирован
1	8	Зарезервирован
2	8	Используется для обмена с НГМД
3	8	Используется для обмена с НЖМД
4	16	Используется портом параллельного интерфейса
5	16	Зарезервирован
6	16	Зарезервирован
7	16	Зарезервирован



а) традиционный способ б) способ с использованием ПДП

Рис. В.1 Обмен информацией между внешним устройством и ОЗУ

Логически контроллер ПДП разделен на две части (в ранних моделях IBM PC использовались два каскадно соединенных контроллера - 8-битовый и 16-битовый), поэтому для программирования 8-битовых и 16-битовых каналов до сих пор используются разные адреса портов ввода-вывода (см. табл. В.2).

Таблица В.2

Регистры контроллера ПДП

Порты для каналов 0-3	Порты для каналов 4-7	Описание
0..7	0C0h-0DFh	0,2,4,6 - младшие (0..15) биты базового адреса данных для каналов; 1,3,5,7 - счетчики в каналах.
8	0D0h	Регистр управления/состояния.
9	0D2h	Регистр программного запроса.
0Ah	0D4h	Регистр маски запросов. Биты 0..1 - номер канала; бит 2 - установить/сбросить (0/1) маску.
0Bh	0D6h	Регистр установки режима. Биты 0..1 - номер канала; биты 2..3: 00-проверка; 01-запись на

		устройство; 10-чтение с устройства; бит 4: 1 - автоинициализация ; бит 5: 0 - инкремент адреса, 1- декремент; биты 6-7: 00 - передача по запросу, 01 - одиночная передача, 10 - блочная передача.
0Ch	0D8h	Регистр сброса триггера.
0Dh	0DAh	Регистр сброса контроллера ПДП.
0Eh	0DCh	Регистр сброса масок.
0Fh	0DEh	Регистр маскирования каналов. Биты 0-3 - маскирование/размаскирование (1/0) каналов.
81h-8Fh	-“-	Старшие (16..19) биты базового адреса: каналу 1 соответствует порт 83h.

При работе в режиме ПДП предполагается, что для обмена данными с внешним устройством в оперативной памяти ПЭВМ выделен фрагмент, для которого известны длина в байтах и базовый **линейный** адрес (т.е. адрес, представленный в виде одного 20-битового числа, вычисляемого как СЕГМЕНТ*16+СМЕЩЕНИЕ). **Фрагмент не должен пересекать границы физического сегмента памяти!** Инициализация канала контроллера ПДП может быть выполнена следующим образом.

Шаг 1. Замаскировать запросы посылкой в порты 0Ah или 0D4h значения 000001kk (здесь и далее kk - номер канала по модулю 4, т.е. канал 4 кодируется значением 0, канал 5 значением 1 и т.д.).

Шаг 2. Сбросить триггер посылкой в порт 0Ch или 0D8h значения 0.

Шаг 3. Установить режим работы контроллера ПДП посылкой в порт 0Bh или 0D6h значения 010a10kk (вывод на внешнее устройство) или 010a01kk (ввод со внешнего устройства). Если бит "a" установлен в 1, то после выполнения операции ввода-вывода контроллер ПДП произведет автоинициализацию в прежнем режиме.

Шаг 4. Загрузить в контроллер младшие 16 битов базового адреса, сначала младший байт, потом старший. Загрузить старшие 4 бита базового адреса.

Шаг 5. Загрузить значение счетчика (количество передаваемых байтов - 1), сначала младший байт, потом старший.

Шаг 6. Размаскировать запросы посылкой в порты 0Ah или 0D4h значения 000000kk.

Приложение В. Описания некоторых инструментов для AdLib

Регистры инстр. 1	Регистры инстр. 2	Фоно	Зуммер	Гитара
0020	0021	0001	0001	0001
0040	0041	0011	0010	004f
0060	0061	00f2	00F0	00f1
0080	0081	00ef	0077	00ad
00e0	00e1	0000	0000	0000
0023	0024	0001	0001	0011

0043	0044	0000	0000	0000
0063	0064	00f5	00F0	00a4
0083	0084	0078	0077	008e
00e3	00e4	0000	0000	0000
00c0	00c1	000a	0000	0006

Приложение Г. Формат WAV-файла

Файл с расширением .WAV разработан и используется фирмой Microsoft для хранения звуковой информации. Информация представлена в виде множества выборочных значений - числовых кодов уровня звукового сигнала.

Файл состоит из одного общего заголовка длиной 12 байтов, начинающегося сигнатурой «RIFF» и кончающегося сигнатурой «WAVE», и некоторого количества секций, содержащих звуковые данные и служебную информацию.

Каждая такая секция также начинается с заголовка, вслед за которым непосредственно располагается содержимое секции. Заголовок каждой секции состоит из уникальной сигнатуры (текстовой строки длиной 4 байта) и 32-битового слова длины содержимого. По сигнатуре можно определить тип секции: «data» - собственно звуковые данные, «fmt » - описание формата звуковых данных, «fact» - вспомогательные сведения о файле, «cue» - адреса отдельных звуковых фрагментов внутри файла и т.п.

В наиболее простом случае WAV-файл состоит из общего заголовка, секции формата и секции данных (см. рис. Г.1). Данные располагаются непосредственно после описанной части.

Таблица Г.1

Структура WAV-файла

Смещ.	Длина	Описание
0	4	‘RIFF’ – начало заголовка файла
4	4	Длина файла без заголовка
8	4	‘WAVE’ – конец заголовка файла
12	4	‘fmt ’ – начало заголовка секции формата
16	4	Длина секции формата
20	2	Тип данных: 1 - непосредственные значения уровня звука; иначе - сжатые или закодированные данные.
22	2	Число каналов (моно, стерео, квадро и т.п.)
24	4	Частота дискретизации звукового сигнала в Гц
28	4	Рекомендуемая частота передачи
32	2	Параметр выравнивания
34	2	Число битов на один отсчет (8, 16 и т.п.)
36	4	‘data’ – начало заголовка секции данных
40	4	Длина секции данных
44	-	Собственно звуковые данные

Приложение Д. Примеры программирования звука на ПЭВМ

Большинство приведенных примеров используют непосредственную работу с портами ввода-вывода ПЭВМ. В операционных системах типа Windows NT/2000/XP и т.п. доступ к ним возможен с использованием специальных драйверов типа PortIO, IOMap, TwicHW32 и пр.

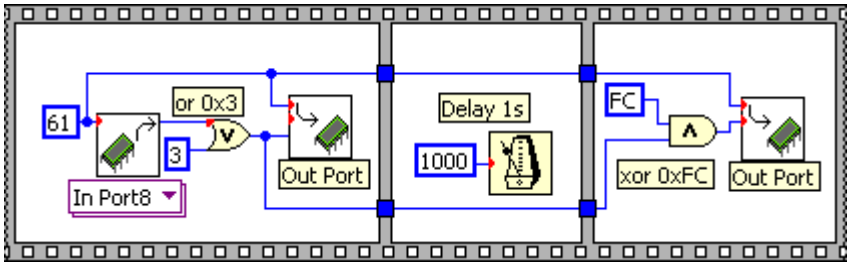
```
(*****************************************************************)
(* Пример программирования встроенного динамика, язык Modula-2   *)
(*****************************************************************)
MODULE M2PIANO;

IMPORT Lib, SYSTEM;

PROCEDURE Note(F, T: CARDINAL); (*Нота с частотой F длительностью T*)
 VAR TMP : SYSTEM.Registers;
BEGIN
 SYSTEM.Out(43H, 0B6H);(*Установка режима*)
 TMP.AX:=CARDINAL(1193180 DIV LONGCARD(F)); (*Расчет константы*)
 SYSTEM.Out(42H,TMP.AL); (*Загрузка младшего байта*)
 SYSTEM.Out(42H,TMP.AH); (*Загрузка старшего байта*)
 SYSTEM.Out(61H, 3);   (*Разрешение звучания*)
 Lib.Delay(T);         (*Задержка*)
 SYSTEM.Out(61H,0);    (*Запрет звучания*)
END Note;

BEGIN  (*Низкочастотная гамма*)
 Note(65, 200);
 Note(72, 200);
 Note(82, 200);
 Note(87, 200);
 Note(98, 200);
 Note(110, 200);
 Note(127, 200);
END M2PIANO.
```

```
/*****************************************************************/
/*               Пример программирования AdLib, язык Си          */
/*****************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>

/*Запись данных "c" в регистр "r" */
void write_fm(unsigned char r, unsigned char c) {
 outportb(0x388,r); delay(1); outportb(0x389,c); delay(1);
}

/*Проверка наличия AdLib*/
int detect_fm() {
 write_fm(4, 0x60); write_fm(4, 0x80);
 if (inportb(0x388)&0xE0) return 0;
 write_fm(2, 0xFF); write_fm(4, 0x21);
 if ((inportb(0x388)&0xC0)!=0xC0) return 0;
 return 1;
}

/*Процедура установки ноты "n" для октавы "o" */
void set_note(unsigned char r, int n, unsigned char o){
```

```
 write_fm(r, n&0xFF);
 write_fm(r+0x10, (n>>8)|(o<<2)|0x20);
}

main() {
 int i;
 if (detect_fm()) { /*Проверить наличие АдЛиба*/
    /*Сбросить AdLib*/
    for (i=0;i<244;i++) write_fm(i, 0);
    /* Проинициализировать AdLib для инструмента "Зуммер" */
    write_fm(0x20, 1);
    write_fm(0x40, 0x10);
    write_fm(0x60, 0xf0);
    write_fm(0x80, 0x77);
    write_fm(0xC0, 0x0);
    write_fm(0x23, 1);
    write_fm(0x43, 0x0);
    write_fm(0x63, 0xf0);
    write_fm(0x83, 0x77);
    /* Гамма 4 октавы */
    set_note(0xA0, 0x181, 4); // Re
    delay(1000);
    set_note(0xA0, 0x1B0, 4); // Mi
    delay(1000);
    set_note(0xA0, 0x1CA, 4); // Fa
    delay(1000);
    set_note(0xA0, 0x202, 4); // Sol
    delay(1000);
    set_note(0xA0, 0x241, 4); // La
    delay(1000);
    set_note(0xA0, 0x287, 4); // Si
    delay(1000);
    set_note(0xA0, 0x2AE, 4); // Do
    delay(1000);
    /* Выключить звук */
    write_fm(0xB0, 0);
    /*Сбросить AdLib*/
    for (i=0;i<244;i++) write_fm(i, 0);
 }
}

(*************************************************************)
(*    Пример программирования саундбластера, язык Паскаль    *)
(*************************************************************)
program paspiano(input, output);

uses Crt;

var
 BASE  : integer;  {Базовый адрес саундбластера}
 Found : boolean;  {Признак присутствия саундбластера}

procedure dspwr(b:byte); {Запись байта в DSP}
begin
 repeat until (port[BASE+$0C] and $80)=0;
 port[BASE+$0C]:=b;
end;

procedure dsprd(var b:byte); {Чтение байта из DSP}
begin
 repeat until (port[BASE+$0E] and $80)<>0;
 b:=port[BASE+$0A];
end;
```

```
procedure nota(f, d:integer); {Нота частотой ”f” длительностью ”d” мс}
 var t:integer;
begin
  t:=0;
  while t<=d do begin
   {Прямоугольная волна}
   dspwr($10);       (* Команда записи               *)
   dspwr(64);        (* Вывод на ЦАП высокого уровня *)
   delay(500 div f); (* Задержка на 1-ю полуволну    *)
   dspwr($10);       (* Команда записи               *)
   dspwr(192);       (* Вывод на ЦАП низкого  уровня *)
   delay(500 div f); (* Задержка на 2-ю полуволну    *)
   t:=t + (1000 div f); (* Считаем длительность      *)
  end;
end;

begin
 (*Поиск и инициализация саундбластера*)
 BASE:=$220; Found:=FALSE;
 while (BASE<$270) and (not Found) do begin
  port[BASE+$06]:=1;
  delay(1);
  port[BASE+$06]:=0;
  delay(1);
  if port[BASE+$0A]=$AA then Found:=TRUE else BASE:=BASE+$10;
 end;
 if Found then begin
  dspwr($D1);     (* Разрешение звука *)
  nota(65,500);   (* Низкочастотная гамма *)
  nota(72,500);
  nota(82,500);
  nota(87,500);
  nota(98,500);
  nota(110,500);
  nota(127,500);
 end
end.
////////////////////////////////////////////////////////////////////////////
// Пример программирования под Windows (включить библиотеку Winmm.lib)       //
////////////////////////////////////////////////////////////////////////////
#include <windows.h>
#include <mmsystem.h>
HMIDIOUT h;
main() {
// Демонстрация проигрывания WAV-файла
  sndPlaySound("c:\\winnt\\media\\tada.wav",SND_SYNC);  Sleep(1000);
// Демонстрация доступа к MIDI, проигрывание гаммы
  midiOutOpen(&h, MIDI_MAPPER, 0, 0, 0);
  midiOutShortMsg(h, 0x00000DC0);
  midiOutShortMsg(h, 0x007F3C90); Sleep(500); midiOutShortMsg(h, 0x007F4680);
  midiOutShortMsg(h, 0x007F3E90); Sleep(500); midiOutShortMsg(h, 0x007F4680);
  midiOutShortMsg(h, 0x007F4090); Sleep(500); midiOutShortMsg(h, 0x007F4680);
  midiOutShortMsg(h, 0x007F4290); Sleep(500); midiOutShortMsg(h, 0x007F4680);
  midiOutShortMsg(h, 0x007F4490); Sleep(500); midiOutShortMsg(h, 0x007F4680);
  midiOutShortMsg(h, 0x007F4690); Sleep(500); midiOutShortMsg(h, 0x007F4680);
  midiOutShortMsg(h, 0x007F4890); Sleep(500); midiOutShortMsg(h, 0x007F4680);
  midiOutShortMsg(h, 0x007F4A90); Sleep(500); midiOutShortMsg(h, 0x007F4680);
  midiOutShortMsg(h, 0x007F4C90); Sleep(500); midiOutShortMsg(h, 0x007F4680);
  midiOutShortMsg(h, 0x007F4E90); Sleep(500); midiOutShortMsg(h, 0x007F4680);
  midiOutShortMsg(h, 0x007F5090); Sleep(500); midiOutShortMsg(h, 0x007F4680);
  midiOutShortMsg(h, 0x007F5290); Sleep(500); midiOutShortMsg(h, 0x007F4680);
  midiOutClose(h);
}
```

Пример доступа к встроенному динамику через порты ввода-вывода в среде LabView. Панель блок-диаграммы содержит стандартные элементы «Out Port» («Вывод в порт») и «In Port» («Ввод из порта») и «Wait Until Next ms Multiple» («Ожидания начала следующей миллисекунды»). Виртуальный прибор прочитывает из порта 0x61 байт, устанавливает в нем два младших бита (логическим сложением с 0x3) и возвращает в порт. В течение 1с звучит сигнал той частоты, которая была установлена средствами BIOS.POST при загрузке компьютера. По истечении секунды виртуальный прибор сбрасывает в байте два младших бита (логическим умножением на 0xFC) и вновь отправляет в порт – звучание прекращается.



Литература

1. Зубков С.В. Assembler. Для DOS, Windows и Unix. - М.: ДМК, 1999. - 640 с.
2. Фролов А.В., Фролов Г.В. Аппаратное обеспечение IBM PC: В 2-х ч. Ч. 2. - М.: ДИАЛОГ-МИФИ, 1992. - 208 с.
3. Гордеев О. Программирование звука в Windows. – СПб: BHV-Санкт-Петербург, 2000. – 384 с.