Аппаратное обеспечение персонального компьютера

© Александр Фролов, Григорий Фролов
Том 33, М.: Диалог-МИФИ, 1997, 304 стр.

5 Системный таймер

- Обработка прерываний таймера
- Микросхемы таймера 8253 и 8254
- Программирование таймера на уровне портов
- Средства BIOS для работы с таймером
- Средства MS-DOS для работы с таймером
- Таймер и музыка

Кроме часов реального времени, любой компьютер (даже простейший IBM PC) содержит устройство, называемое системным таймером. Это устройство подключено к линии запроса на прерывание IRQ0 и вырабатывает прерывание INT 8h приблизительно 18,2 раза в секунду (точное значение - 1193180/65536 раз в секунду).

Обработка прерываний таймера

При инициализации BIOS устанавливает свой обработчик для прерывания таймера. Этот обработчик каждый раз увеличивает на единицу текущее значение 4-байтовой переменной, располагающейся в области данных BIOS по адресу 0000:046Ch - счетчик таймера. Если этот счетчик переполнится из-за того что прошло более 24 часов с момента запуска таймера, в ячейку 0000:0470h заносится значение 1.
Другое действие, выполняемое стандартным обработчиком прерывания таймера - контроль за работой двигателей НГМД. Если после последнего обращения к НГМД прошло более 2 секунд, обработчик прерывания выключает двигатель. Ячейка с адресом 0000:0440h содержит время, оставшееся до выключения двигателя. Это время постоянно уменьшается обработчиком прерывания таймера. Когда оно становится равно 0, двигатель НГМД отключается.
Последнее действие, которое выполняет обработчик прерывания таймера - вызов прерывания INT 1Ch. После инициализации системы вектор INT 1Ch указывает на команду IRET, то есть обработчик прерывания INT 1Ch ничего не делает. Программа может установить собственный обработчик этого прерывания для того чтобы выполнять какие-либо периодические действия.

Необходимо отметить, что прерывание INT 1Ch вызывается обработчиком прерывания INT 8h до сброса контроллера прерывания, поэтому во время выполнения прерывания INT 1Ch все аппаратные прерывания запрещены. В частности, запрещены прерывания от клавиатуры.
Обработчик прерывания INT 1Ch должен заканчиваться командой IRET.
Если же вы подготавливаете собственный обработчик для прерывания INT 8h, перед завершением его работы необходимо сбросить контроллер прерываний. Это можно сделать, например, так:

```
mov  al, 20h
out  20h, al
```

Таймер обычно реализуется на микросхеме Intel 8253 (для компьютеров IBM PC и IBM PC/XT) или 8254 (для компьютеров IBM PC/AT и IBM PS/2), а также на аналогах этих микросхем. Следующий раздел книги посвящен описанию микросхемы 8254.
Мы не будем подробно рассказывать о всех возможностях микросхем 8253 и 8254, так как обычно используются только несколько режимов работы (а чаще всего один). Полное описание вы сможете найти в справочной литературе по микросхемам 8253/8254, а также по их отечественным аналогам КР1810ВИ53 и КР1810ВИ54.

Микросхемы таймера 8253 и 8254

Таймеры 8253 и 8254 состоят из трех независимых каналов, или счетчиков. Каждый канал содержит регистры:

- состояния канала RS (8 разрядов);
- управляющего слова RSW (8 разрядов);
- буферный регистр OL (16 разрядов);
- регистр счетчика CE (16 разрядов);
- регистр констант пересчета CR (16 разрядов)

Каналы таймера подключаются к внешним устройствам при помощи трех линий:

- GATE - управляющий вход;
- CLOCK - вход тактовой частоты;
- OUT - выход таймера

Регистр счетчика CE работает в режиме вычитания. Его содержимое уменьшается по заднему фронту сигнала CLOCK при условии, что на вход GATE установлен уровень логической 1.
В зависимости от режима работы таймера при достижении счетчиком CE нуля тем или иным образом изменяется выходной сигнал OUT.
Буферный регистр OL предназначен для запоминания текущего содержимого регистра счетчика CE без остановки процесса счета. После запоминания буферный регистр доступен программе для чтения.

Регистр констант пересчета CR может загружаться в регистр счетчика, если это требуется в текущем режиме работы таймера.
Как нетрудно догадаться по названию, регистры состояния канала и управляющего слова предназначены, соответственно, для определения текущего состояния канала и для задания режима работы таймера.

Режимы работы таймера

Возможны шесть режимов работы таймера. Они разделяются на три типа:

- режимы 0, 4 - однократное выполнение функций.
- режимы 1, 5 - работа с перезапуском.
- режимы 2, 3 - работа с автозагрузкой

Режим однократного выполнения функций

В режиме однократного выполнения функций перед началом счета содержимое регистра констант пересчета CR переписывается в регистр счетчика CE по сигналу CLOCK, если сигнал GATE установлен в 1. В дальнейшем содержимое регистра CE уменьшается по мере прихода импульсов CLOCK. Процесс счета можно приостановить, если подать на вход GATE уровень логического 0. Если затем на вход GATE подать 1, счет будет продолжен дальше. Для повторения выполнения функции необходима новая загрузка регистра CR, то есть повторное программирование таймера.

Работа с перезапуском

При работе с перезапуском не требуется повторного программирования таймера для выполнения той же функции. По фронту сигнала GATE значение константы из регистра CR вновь переписывается в регистр CE, даже если текущая операция не была завершена.

Режим автозагрузки

В режиме автозагрузки регистр CR автоматически переписывается в регистр CE после завершения счета. Сигнал на выходе OUT появляется только при наличии на входе GATE уровня логической 1. Этот режим используется для создания программируемых импульсных генераторов и генераторов прямоугольных импульсов (меандра).

Каналы таймера

В соверменных компьютерах задействованы все три канала таймера.

Канал 0

Канал 0 используется в системных часах времени суток (не следует путать с часами реального времени, реализованными на другой микросхеме). Этот канал работает в режиме 3 и используется как генератор импульсов с

частотой примерно 18,2 Гц. Именно эти импульсы вызывают аппаратное прерывание INT 8h.

Канал 1

Канал 1 используется для регенерации содержимого динамической памяти компьютера. Выход канала OUT используется для запроса к каналу прямого доступа DMA, который и выполняет обновление содержимого памяти. Вам не следует перепрограммировать этот канал, так как это может привести к нарушениям в работе основной оперативной памяти компьютера.

Канал 2

Канал 2 подключен к громкоговорителю компьютера и может быть использован для генерации различных звуков или музыки, либо как генератор случайных чисел. Канал использует режим 3 таймеров 8253 и 8254.

Программирование таймера на уровне портов

Для чего вам может понадобиться перепрограммирование каналов таймера?
Если вам надо повысить точность измерения времени, выполняемого с помощью канала 0 таймера, вы можете увеличить частоту генерируемых этим каналом импульсов (стандартно 18,2 Гц). По окончании измерений режим работы канала необходимо восстановить для правильной работы системы.
Канал 2, подключенный к громкоговорителю, вы можете использовать для генерации различных звуков или музыки, о чем мы расскажем немного позже. Этот же канал пригодится и для генерации случайных чисел.
Таймеру соответствуют четыре порта ввода/вывода со следующими адресами:

- 40h - канал 0;
- 41h - канал 1;
- 42h - канал 2;
- 43h - управляющий регистр

Формат управляющего регистра

Приведем формат управляющего регистра:

Поля регистра	Описание
0	Поле BCD: 0 - двоичный режим; 1 - двоично-десятичный режим
1-3	Поле M:

	000 - режим 0; 001 - режим 1; X10 - режим 2; X11 - режим 3; 100 - режим 4; 101 - режим 5
4-5	Поле RW: 00 - код команды CLC (запомнить CE); 01 - чтение/запись старшего байта; 10 - чтение/запись младшего байта; 11 - чтение/запись младшего, затем старшего байта
6-7	Поле SC: 00 - канал 0; 01 - канал 1; 10 - канал 2; 11 - код команды RBC (чтение состояния канала)

Поле BCD определяет формат константы, использующейся для счета - двоичный или двоично-десятичный. В двоично-десятичном режиме константа задается в диапазоне 1-9999.

Поле M определяет режимы работы таймера:

- 0 - прерывание от таймера;
- 1 - программируемый ждущий мультивибратор;
- 2 - программируемый генератор импульсов;
- 3 - генератор меандра;
- 4 - программно-запускаемый одновибратор;
- 5 - аппаратно-запускаемый одновибратор

Мы будем рассматривать только режим 3, так как именно он используется в каналах 0 и 2.

Поле RW определяет способ загрузки констант через однобайтовый порт. Если в этом поле задано значение 00, это управляющее слово будет использоваться для фиксации текущего содержимого регистров счетчика CE в буферном регистре OL с целью чтения программой. Это код команды CLC - фиксация регистров. Код канала, для которого будет выполняться фиксация, должен быть указан в поле SC. Поля M и BCD при этом не используются.

Поле SC определяет номер канала, для которого предназначено управляющее слово. Если в этом поле задано значение 11, будет выполняться чтение состояния канала.

Формат команды чтения слова состояния канала

С помощью команды чтения слова состояния канала вы можете выполнять операции чтения состояния каналов либо запоминание регистра счетчика СЕ каналов. Можно выполнять эти операции как для отдельных каналов, так и для всех каналов одновременно .

Приведем формат команды RBC чтения слова состояния канала:

Поле	Описание
0	Всегда равно 0
1	1 - выбор канала 0
2	1 - выбор канала 1
3	1 - выбор канала 2
4	Поле STAT: 0 - читать состояние каналов; 1 - не читать состояние каналов
5	Поле CNT: 0 - запомнить текущее содержимое СЕ; 1 - не запоминать содержимое СЕ
6-7	код команды RBC - 11

Формат слова состояния канала

Формат слова состояния канала напоминает формат регистра управляющего слова, за исключением двух старших разрядов 7 и 6:

Поля регистра	Описание
0	Поле BCD: 0 - двоичный режим; 1 - двоично-десятичный режим
1-3	Поле M: 000 - режим 0; 001 - режим 1; X10 - режим 2; X11 - режим 3; 100 - режим 4; 101 - режим 5
4-5	Поле RW: 00 - код команды CLC (запомнить СЕ); 01 - чтение/запись старшего байта; 10 - чтение/запись младшего байта; 11 - чтение/запись младшего, затем старшего байта
6	FN: флаг перезагрузки констант

7	OUT: состояние выхода OUT

Разряд FN используется, в основном, в режимах 1 и 5 для определения, произошла ли загрузка константы из регистра CR в регистр счетчика CE. Разряд OUT позволяет определить состояние выходной линии канала OUT в момент выполнения команды RBC.

Последовательность действий

Для программирования канала таймера необходимо выполнить следующую последовательность действий:

· вывести в порт управляющего регистра с адресом 43h управляющее слово;

· требуемое значение счетчика посылается в порт канала (адреса 40h-42h), причем вначале выводится младший, а затем старший байты значения счетчика.

Сразу после этого канал таймера начнет выполнять требуемую функцию. Для чтения текущего содержимого счетчика CE необходимо выполнить следующее:

· вывести в порт управляющего регистра код команды CLC (команда запоминания содержимого регистра CE);

· вывести в порт управляющего регистра код команды запроса на чтение/запись в регистры канала (поле RW должно содержать 11);

· двумя последовательными командами ввода из порта нужного канала ввести младший и старший байты текущего сосотояния счетчика CE.

Программа TIMERST

Приведем исходный текст программы TIMERST, отображающей слово состояния и содержимое счетчика для всех трех каналов таймера (листинг 5.1).

Листинг 5.1. Файл timerst\timerst.c

```
        //
=============================================================
        // Просмотр слова состояния и содержимого
        // счетчиков таймера
        //
        // (C) Фролов А.В, 1997
        //
        // E-mail: frolov@glas.apc.org
        // WWW:    http://www.glasnet.ru/~frolov
        //           или
        //        http://www.dials.ccas.ru/frolov
        //
=============================================================
```

```
#include <stdio.h>
#include <conio.h>

int main()
{
  unsigned i;

  printf("\n\nКанал 0\n-------\n");

  // Читаем слово состояния канала,
  // команда 0xe2 = 11100010B
  outp(0x43, 0xe2);

  printf("\nСлово состояния канала: %02.2X",
    inp(0x40));

  // Читаем текущее состояние регистра счетчика
  // канала. Для этого вначале выдаем команду CLC
  // для канала 0. Код этой команды - 0x00
  outp(0x43, 0x00);

  // Вводим младший и старший байты счетчика
  // и отображаем его.
  i = inp(0x40);
  i = (inp(0x40) << 8) + i;

  printf("\nРегистр счетчика:      %04.4X",i);

  // Повторяем те же действия для 1 и 2 каналов.
  printf("\n\nКанал 1\n-------\n");
  outp(0x43, 0xe4);
  printf("\nСлово состояния канала: %02.2X",inp(0x41));
  outp(0x43, 0x40);
  i = inp(0x41);
  i = (inp(0x41) << 8) + i;
  printf("\nРегистр счетчика:      %04.4X",i);

  printf("\n\nКанал 2\n-------\n");
  outp(0x43, 0xe8);
  printf("\nСлово состояния канала: %02.2X",inp(0x42));
  outp(0x43, 0x80);
  i = inp(0x42);
  i = (inp(0x42) << 8) + i;
  printf("\nРегистр счетчика:      %04.4X",i);
  return 0;
}
```

Средства BIOS для работы с таймером

Для работы с таймером (точнее говоря, для работы с каналом 0 таймера) BIOS содержит две функции прерывания INT 1Ah и две функции

прерывания INT 15h. Они позволяют прочитать текущее содержимое счетчика и изменить его, установить таймер с сигнализацией и сформировать программную задержку.

Чтение счетчика таймера

Функция 00h предназначена для чтения содержимого счетчика таймера:

Регистры на входе:	AH = 00h
Регистры на выходе:	CX = старший байт счетчика; DX = младший байт счетчика; AL = 0, если с момента перезапуска таймера прошло более 24 часов

Установка счетчика таймера

Изменить содержимое счетчика таймера можно с помощью следующей функции:

Регистры на входе:	AH = 01h CX = старший байт счетчика; DX = младший байт счетчика
Регистры на выходе:	Регистры не используются

Функцию чтения таймера можно использовать для организации программной задержки. Так как работа таймера не зависит от производительности процессора, быстродействие системы не будет влиять на формируемую задержку.
Однако следует учитывать, что точность формирования задержки определяется частотой обновления счетчика таймера (18.2 Гц), и может оказаться недостаточной для некоторых приложений.

Установка таймера с сигнализацией

BIOS компьютеров IBM PC/AT содержит еще две интересные функции для работы с таймером. Это функции 83h и 86h прерывания INT 15h.
Функция 83h INT 15h позволяет запустить таймер на счет, указав адрес некоторого байта в оперативной памяти. Программа, запустившая таймер, сразу после запуска получает управление. По истечении времени, заданного при запуске таймера, функция устанавливает старший бит указанного байта в единицу, сигнализируя таким образом программе о завершении указанного временного интервала. Программа может также отменить работу таймера в этом режиме.
Эту функцию удобно использовать для организации выполнения каких-либо действий параллельно с отсчетом времени, например, можно ограничить время для ввода пароля.

Приведем формат вызова функции 83h прерывания INT 15h:

Регистры на входе:	AH = 83h AL = код подфункции: 0 – установить интервал, запустить таймер; 1 - отменить работу таймера; CX = старший байт времени работы счетчика, задается в микросекундах; DX = младший байт счетчика; ES:BX = адрес байта, в котором по истечении интервала времени будет установлен старший бит
Регистры на выходе:	Регистры не используются

Формирование задержки

Функция 86h специально предназначена для формирования задержек. Она позволяет определять время задержки в микросекундах, что достаточно удобно для многих задач. Во время выполнения задержки разрешены прерывания.

Формат вызова функции:

Регистры на входе:	AH = 86h CX = старшее слово времени работы счетчика, задается в микросекундах; DX = младшее слово счетчика;
Регистры на выходе:	Регистры не используются

Средства MS-DOS для работы с таймером

MS-DOS использует четыре функции прерывания INT 21h для работы с системным таймером. Эти функции позволяют узнать и установить текущие дату и время. MS-DOS версии 3.30 и более поздних версий при установке времени и даты изменяет также показания часов реального времени.

Определение текущей даты

Для определения текущей даты используется функция 2Ah:

Регистры на входе:	AH = 2Ah
Регистры на выходе:	DL = день (0...31); DH = месяц (1...12); CX = год (1980...2099); AL = номер дня недели: 0 - воскресенье;

	1 - понедельник; 2 - вторник; . . . 6 - суббота

Обратите внимание на то, что функция возвращает вам номер дня недели, который она вычисляет на основе даты.

Установка даты

Для установки даты используйте функцию 2Bh:

Регистры на входе:	AH = 2Bh DL = день (0...31); DH = месяц (1...12); CX = год (1980...2099)
Регистры на выходе:	AL = 0, если установка выполнена правильно; AL = FFh, если при установке были заданы неправильные параметры

Определение текущего времени

Для того, чтобы определить текущее время, можно воспользоваться функцией 2Ch:

Регистры на входе:	AH = 2Ch
Регистры на выходе:	CH = часы (0-24); CL = минуты (0-59); DH = секунды(0...59); DL = сотые доли секунды (0-99)

Точность времени, полученного при помощи этой функции, определяется таймером (его счетчик обновляется 18,2 раза в секунду).

Установка времени

Для установки времени можно использовать функцию 2Dh:

Регистры на входе:	AH = 2Dh CH = часы (0-24); CL = минуты (0-59); DH = секунды(0-59); DL = сотые доли секунды (0-99)
Регистры на выходе:	AL = 0, если установка выполнена правильно; AL = FFh, если при установке были заданы неправильные параметры

Функции стандратной библиотеки С

Стандартные библиотеки С содержат многочисленные функции для работы с датой и временем. Они основаны на описанных выше функциях MS-DOS и предоставляют широкие возможности для отображения даты и времени в различных форматах. Подробное описание этих функций и примеры их использования вы найдете в документации на библиотеку С.

Таймер и музыка

Одно из наиболее распространенных применений таймера - генерация звуковых сигналов и проигрывание музыки. Таймер позволяет проигрывать музыку в фоновом режиме, то есть во время работы программы может звучать музыка.

Настройка таймера для проигрывания музыки

Как мы уже говорили, канал 2 микросхемы 8254 связан с громкоговорителем компьютера. Однако громкоговоритель не просто соединен с выходом OUT канала 2. Порт вывода 61h также используется для управления громкоговорителем. Младший бит порта 61h подключен ко входу GATE канала 2 таймера. Этот бит при установке в 1 разрешает работу канала, то есть генерацию импульсов для громкоговорителя. Дополнительно для управления громкоговорителем используется бит 1 порта 61h. Если этот бит установлен, импульсы от канала 2 таймера смогут проходить на громкоговоритель.

Таким образом, для включения звука надо выполнить следующие действия:

· запрограммировать канал 2 таймера на нужную частоту (загрузить регистр счетчика канала нужным значением);

· для включения звука установить два младших бита порта 61h.

Так как остальные 6 битов порта 61h используются для других целей, установка младших битов должна выполняться таким образом, чтобы значения остальных битов не были изменены. Для этого вначале надо считать байт из порта 61h в рабочую ячейку памяти, установить там нужные биты, затем вывести новое значение байта в порт 61h.

Для выключения звука надо сбросить два младших бита порта 61h, при этом нельзя изменять значение остальных битов этого порта.

Одноголосая мелодия состоит из нот, разделенных или не разделенных паузами. При проигрывании мелодии необходимо для каждой ноты программировать соответствующим образом канал 2 таймера и включать громкоговоритель (с помощью порта 61h) на определенное время, равное длительности ноты. Затем программа должна выключить динамик и выдержать паузу перед проигрыванием следующей ноты, если такая пауза требуется.

Второй способ проигрывания музыки

Программа может генерировать звуки и другим способом, не используя таймер. Для этого нужно сбросить младший бит порта 61h и, управляя битом 1 этого порта, формировать импульсы для громкоговорителя. Иными словами, программа должна периодически то устанавливать этот бит, то сбрасывать. Высота генерируемого звука будет соответствовать периоду изменения состояния указанного бита.
Можно также комбинировать описанные способы, получая разнообразные звуковые эффекты.

Проигрывание музыки в фоновом режиме

Для проигрывания мелодии в фоновом режиме можно предложить следующий способ, основанный на использовании периодического прерывания от канала 0 таймера.
Основная идея заключается в использовании прерывания INT 1Ch, которое вырабатывается таймером с частотой примерно 18,2 Гц. Ваш обработчик этого прерывания осуществляет контроль за выборкой нот из массива, содержащего мелодию, и программирование микросхемы 8254. Например, один раз в полсекунды обработчик проверяет, не пора ли прекратить звучание одной ноты и начать проигрывание следующей ноты. Если пора, он выключает громкоговоритель и перепрограммирует канал 8254 на новую частоту, соответствующую следующей ноте.
Основное преимущество использования таймера для проигрывания мелодии - независимость констант, используемых для программирвания канала таймера от производительности системы. Ваша мелодия будет звучать одинаково и на медленной IBM PC/XT и на совеменном компьютера с процессором Pentium, но при условии, что вы будете использовать таймер и для организации задержек при исполнении мелодии.
Для определения значения, которое должно быть записано в регистр счетчика канала 2 таймера, надо разделить число 1193180 на частоту ноты в герцах.
Для подготовки таблиц частот по нотам вам поможет список частот для нот второй октавы:

Нота	Частота, Гц
До	261,7
До-диез	277,2
Ре	293,7
Ре-диез	311,1
Ми	329,6
Фа	349,2

Фа-диез	370,0
Соль	392,0
Соль-диез	415,3
Ля	440,0
Ля-диез	466,2
Си	493,9

Для других октав при понижении или повышении тона значения частот надо умножать (при повышении тона) или делить (при понижении тона) на 2.

Программа TMSOUND

Программа TMSOUND (листинг 5.2) проигрывает мелодию с помощью системного таймера.

Листинг 5.2. Файл tmsound\tmsound.c

```
        //
=============================================================
====
        // Проигрывание музыки с помощью таймера
        //
        // (C) Фролов А.В, 1997
        //
        // E-mail: frolov@glas.apc.org
        // WWW:    http://www.glasnet.ru/~frolov
        //           или
        //        http://www.dials.ccas.ru/frolov
        //
=============================================================
====
        #include <stdio.h>
        #include <conio.h>
        #include <dos.h>

        void sound(int, int);
        void tm_sound(int freq, int time);
        void tm_delay(int ticks);

        // Массив частот для мелодии
        int mary[] =
        {
          330, 294, 262, 294, 330, 330, 330,
          294, 294, 294, 330, 392, 392,
          330, 294, 262, 294, 330, 330, 330, 330,
          294, 294, 330, 294, 262, 0
        };

        // Массив длительностей
```

```
int del[] =
{
  5, 5, 5,  5, 5, 5,  10,
  5, 5, 10, 5, 5, 10,
  5, 5, 5,  5, 5, 5,  5, 5,
  5, 5, 5,  5, 20
};

int main(void)
{
  int i;

  for(i=0 ;mary[i] != 0 ;i++)
    tm_sound(mary[i], del[i]);

  return 0;
}

/**
*.Name        tm_sound
*.Title       Формирование тона заданной длительности
*
*.Descr       Эта функция предназначена для генерации
*            на громкоговорителе тона заданной
*            длительности и частоты
*
*.Proto       void tm_sound(int freq, int time);
*
*.Params      int freq - частота в герцах;
*            int time - длительность в периодах работы
*             таймера
**/
void tm_sound(int freq, int time)
{
  int cnt;

  // Задаем режим канала 2 таймера
  outp(0x43, 0xb6);

  // Вычисляем задержку для загрузки в
  // регистр счетчика таймера
  cnt = (int)(1193180L / freq);

  // Загружаем регистр счетчика таймера - сначала
  // младший, затем старший байты
  outp(0x42, cnt & 0x00ff);
  outp(0x42, (cnt &0xff00) >> 8);

  // Включаем громкоговоритель. Сигнал от
  // канала 2 таймера теперь будет проходить
  // на вход громкоговорителя
```

```
  outp(0x61, inp(0x61) | 3);

  // Выполняем задержку.
  tm_delay(time);

  // Выключаем громкоговоритель.
  outp(0x61, inp(0x61) & 0xfc);
}

/**
*.Name        tm_delay
*.Title       Формирование задержки по таймеру
*
*.Descr       Эта функция формирует задержку, используя
*             системный таймер
*
*.Proto       void tm_delay(int ticks)
*
*.Params      int ticks - величина задержки в периодах работы
таймера
**/

void tm_delay(int ticks)
{
  _asm
  {
    push si

    mov  si, ticks
    mov  ah, 0
    int  1ah

    mov  bx, dx
    add  bx, si

delay_loop:

    int  1ah
    cmp  dx, bx
    jne  delay_loop

    pop  si
  }
}
```

Программа IOSOUND

Приведем исходный текст программы IOSOUND, генерирующую звук без использования таймера (листинг 5.3.). Эта программа формирует импульсы при помощи манипуляций с разрядом 1 порта 61h.

Листинг 5.3. Файл iosound\iosound.c

```
        //
=============================================================
====
        // Генерация звукового сигнала через порты таймера
        //
        // (C) Фролов А.В, 1997
        //
        // E-mail: frolov@glas.apc.org
        // WWW:    http://www.glasnet.ru/~frolov
        //           или
        //        http://www.dials.ccas.ru/frolov
        //
=============================================================
====
        #include <stdio.h>
        #include <conio.h>
        #include <dos.h>

        #define FREQUENCY 200
        #define CYCLES 30000

        int main(void)
        {
          // Во время генерации звука прерывания должны
          // быть запрещены.
          _disable();

          _asm
          {
            // Загружаем количество циклов - периодов
            // генерируемых импульсов
            mov   dx, CYCLES

            // Отключаем громкоговоритель от таймера
            in    al, 61h
            and   al, 0feh

            // Цикл формирования периода
        sound_cycle:

            // Формируем первый полупериод, подаем
            // на громкоговоритель уровень 1
            or    al, 2
            out   61h, al

            // Формируем задержку
            mov   cx, FREQUENCY

        first: loop  first

            // Формируем второй полупериод, подаем
```

```
            // на громкоговоритель уровень 0
            and   al,  0fdh
            out   61h, al

            // Формируем задержку
            mov   cx, FREQUENCY

        second: loop  second

            // Если сформированы не все периоды, переходим
            // к формированию следующего периода.
            dec   dx
            jnz   sound_cycle
          }

          // Разрешаем прерывания
          _enable();

          // Выключаем громкоговоритель
          outp(0x61, inp(0x61) & 0xfc);

          return 0;
        }
```

Так как в программе IOSOUND для формирования полупериодов используется задержка с помощью команды LOOP, высота генерируемого тона будет зависеть от производительности системы.
Такой зависимости можно избежать, если перед началом работы измерять производительность и соответствующим образом корректировать константу, загружаемую в регистр CX перед вызовом команды LOOP. Измерение производительности лучше всего выполнять с помощью таймера, определяя время выполнения команды LOOP.

Программа RANDOM

Последнее, что мы сделаем с таймером - научимся получать от него случайные числа.
Для генерации случайных чисел лучше всего использовать канал 2 в режиме 3. В регистр счетчика канала мы занесем значение, равное диапазону нужных нам случайных чисел. Например, если мы запишем в регистр число 80 и запустим канал таймера, получаемые случайные числа будут лежать в диапазоне от 0 до 79.
Программа RANDOM (листинг 5.4) получает случайные числа и отображает их в наглядном виде с помощью столбчатой диаграммы.
Листинг 5.4. Файл random\random.c

```
        //
==========================================================
====
```

```
        // Генерация случайных чисел
        //
        // (C) Фролов А.В, 1997
        //
        // E-mail: frolov@glas.apc.org
        // WWW:    http://www.glasnet.ru/~frolov
        //           или
        //        http://www.dials.ccas.ru/frolov
        //
================================================================
====
        #include <stdio.h>
        #include <conio.h>
        #include <dos.h>

        void rnd_set(int bound);
        int rnd_get(void);

        int main(void)
        {
          int i, j;

          printf("\nГенератор случайных чисел."
            "\nНажмите любую клавишу,"
                "\nдля завершения работы нажмите <Control+C>"
                "\n");

          for(;;)
          {
            // Устанавливаем диапазон генерации случайных
            // чисел и инициализируем таймер
            rnd_set(80);

            // Ожидаем нажатия клавиши
            getch();

            // После нажатия на клавишу получаем
            // случайное число
            j = rnd_get();

            // Выводим на экран строку символов "-",
            // длина которой равна полученному случайному числу
            for(i=0; i < j; i++)
              putchar(219);
            printf("\n");
          }
          return 0;
        }

        /**
        *.Name        rnd_set
```

```
*.Title        Инициализация генератора случайных чисел
*
*.Descr        Эта функция инициализирует канал 2 таймера
*              для использования в качестве генератора
*              случайных чисел
*
*.Proto        void rnd_set(int bound)
*
*.Params       int bound - верхняя граница для генерируемых
*                     случайных чисел.
**/

void rnd_set(int bound)
{
  // Устанавливаем режим 3 для второго канала таймера
  outp(0x43, 0xb6);

  // Загружаем регистр счетчика таймера - сначала
  // младший, затем старший байты
  outp(0x42, bound & 0x00ff);
  outp(0x42, (bound &0xff00) >> 8);

  // Разрешаем работу канала
  outp(0x61, inp(0x61) | 1);
}

/**
*.Name         rnd_get
*.Title        Получение от таймера случайного числа
*
*.Descr        Эта функция получает случайное число от
*              таймера, который был предварительно
*              проинициализирован функцией rnd_set
*
*.Proto        int rnd_get(void)
*
*.Params       Отсутствуют.
*
*.Return       Случайное число в диапазоне от 0, до
*              уменьшенного на 1 значения, заданного в
*              качестве параметра функции rnd_set().
**/

int rnd_get(void)
{
  int i;

  // Выдаем команду CLC для фиксирования
  // текущего значения регистра канала 2 таймера
  outp(0x43, 0x86);
```

```
  // Вводим младший и старший байты счетчика
  i = inp(0x42);
  i = (inp(0x42) << 8) + i;

  return(i);
}
```