ГЛАВА 10

Таймер

В любом компьютере наряду с часами реального времени существует устройство системного таймера, для реализации которого применяется специальная микросхема (например, 8254). Он помогает организовать всевозможные временные задержки, счетчики и управляющие сигналы. Из всех предоставляемых таймером функций, можно выделить несколько основных:

1. Организация часов реального времени.
2. Программируемый генератор прямоугольных и синусоидальных импульсов.
3. Счетчик событий таймера.
4. Управление двигателями флоппи-дисководов.

Таймер предоставляет три независимых канала, каждый из которых имеет свое назначение. В первом канале (0) отслеживается текущее значение времени от момента включения компьютера, для хранения которого используется область памяти BIOS (`0040:006C`). Каждое изменение значения (18,2 раз в секунду) в этом канале генерирует прерывание `IRQ0` (`int 8h`). Данное прерывание будет обработано процессором в первую очередь, но при этом должны быть разрешены аппаратные прерывания. При программировании первого канала всегда следует после выполнения задачи восстанавливать его первоначальное состояние. Второй канал (1) используется системой для работы с контроллером прямого доступа к памяти (DMA). Третий канал (2) позволяет управлять системным динамиком.

Генератор сигналов таймера вырабатывает импульсы с частотой 1 193 180 Гц. Поскольку максимальное значение 16-битного регистра ограничено значением 65 535, используется делитель частоты. В итоге, результирующее значение равно 18,2 Гц. Именно с такой частотой выдается прерывание `IRQ0`.

Для работы с системным таймером используются порты от `40h` до `43h`. Все они имеют размер 8 бит. Порты с номерами `40h`, `41h` и `42h` связаны соответ-

ственно с первым, вторым и третьим каналами, а порт 43h работает с управляющим регистром таймера. Принцип работы с портами очень простой: в порт 43h записывается управляющая команда, а после этого данные записываются или считываются из 40h, 41h и 42h, в зависимости от решаемой задачи. Формат командного регистра представлен в табл. 10.1.

Таблица 10.1. *Формат управляющего регистра таймера*

Биты	7	6	5	4	3	2	1	0
Описание	Номер канала		Тип операции		Режим		Формат	

Приведем краткое описание табл. 10.1.

- Бит 0 определяет формат представления данных: 0 — двоичный (16-битное значение от 0000h до FFFFh), 1 — двоично-десятичный (BCD от 0000 до 9999).
- Биты 1—3 определяют режим работы таймера. Существуют шесть режимов работы (от 0 до 5). Возможные значения перечислены в табл. 10.2.
- Биты 4—5 определяют тип операции. Имеются четыре возможных значения, которые перечислены в табл. 10.3.
- Биты 6—7 позволяют выбрать номер канала или управляющий регистр (только для операций чтения). Возможные значения этого поля перечислены в табл. 10.4.

Таблица 10.2. *Режимы работы таймера*

Бит 3	Бит 2	Бит 1	Описание режима
0	0	0	Генерация прерывания IRQ0 при установке счетчика в 0
0	0	1	Установка в режим ждущего мультивибратора
0	1	0	Установка в режим генератора импульсов
0	1	1	Установка в режим генератора прямоугольных импульсов
1	0	0	Установка в режим программно-зависимого одновибратора
1	0	1	Установка в режим аппаратно-зависимого одновибратора

Таблица 10.3. *Тип операции*

Бит 5	Бит 4	Тип операции
0	0	Команда блокировки счетчика
0	1	Чтение/запись только младшего байта

Таблица 10.3 *(окончание)*

Бит 5	Бит 4	Тип операции
1	0	Чтение/запись только старшего байта
1	1	Чтение/запись младшего, а за ним старшего байта

Таблица 10.4. *Возможные значения для поля 6–7*

Бит 7	Бит 6	Описание
0	0	Выбор первого канала (0)
0	1	Выбор второго канала (1)
1	0	Выбор третьего канала (2)
1	1	Команда считывания значений из регистров каналов

При установке битов 6 и 7 управляющего регистра в 1 будет использована команда считывания данных. При этом формат определения этого регистра меняется согласно табл. 10.5.

Таблица 10.5. *Формат управляющего регистра в режиме считывания*

Биты	7	6	5	4	3	2	1	0
Описание	1	1	Б	Статус	Канал 2	Канал 1	Канал 0	0

Приведем краткое описание таблицы.

- Бит 0 не используется и должен быть установлен в 0.
- Биты 1—3 позволяют установить номера каналов. Установка бита в 1 выбирает соответствующий номер канала.
- Бит 4 позволяет получить состояние для выбранного канала (каналов) при установке в 0.
- Бит 5 позволяет зафиксировать значение счетчика для выбранного канала (каналов) при установке в 0.
- Биты 6 и 7 определяют команду чтения и должны быть установлены в 1.

При выполнении команды блокировки счетчика (биты 4 и 5 установлены в 0) можно получить значение (состояние) выбранного счетчика без остановки самого таймера. Результат считывается из указанного канала (биты 6 и 7). При этом формат команды будет таким, как показано в табл. 10.6.

Таблица 10.6. *Формат команды блокировки*

Биты	7	6	5	4	3	2	1	0
Описание	Номер канала		0		0		Не используются	

Приведем краткий комментарий к таблице.

- Биты 0—3 не используются и должны игнорироваться.
- Биты 4 и 5 определяют команду блокировки и должны быть установлены в 0.
- Биты 6 и 7 определяют номер канала (см. табл. 10.4), для которого будет выполнена команда блокировки.

Полученный байт состояния имеет определенный формат (табл. 10.7).

Таблица 10.7. *Формат байта состояния канала*

Биты	7	6	5	4	3	2	1	0
Описание	OUT	Готов	Тип операции		Режим работы			Формат

Приведем краткое пояснение к таблице 10.7.

- Бит 0 определяет формат представления данных: 0 — двоичный (16-битное значение от `0000h` до `FFFFh`), 1 — двоично-десятичный (BCD от `0000` до `9999`).
- Биты 1—3 определяют режим работы таймера. Возможные значения перечислены в табл. 10.2.
- Биты 4—5 определяют тип операции. Возможные значения перечислены в табл. 10.3.
- Бит 6 определяет готовность счетчика для считывания данных (1 — готов, 0 — счетчик обнулен).
- Бит 7 определяет состояние выходного сигнала на канале в момент блокировки счетчика импульсов.

Рассмотрим стандартный пример для установки начального значения счетчика для второго канала (управляет динамиком) равным 5 120 Гц (листинг 10.1).

Листинг 10.1. Установка начального значения счетчика для второго канала

```
mov AL, 10110110b   ; канал 2, операция 4, режим 3, формат 0
out 43h, AL . ; записываем значение в порт
mov AX, 0E9  ; определяем делитель частоты для получения 5120 Гц
```

```
out 42h, AL  ; посылаем младший байт делителя
mov AL, AH   ; копируем значение
out 42h, AL  ; посылаем старший байт делителя
```

Аналогичный пример для C++ показан в листинге 10.2.

Листинг 10.2. Установка начального значения счетчика для второго канала в C++

```
// пишем функцию установки значения счетчика
void SetCount ( int iDivider)
{
   int iValue = 0;
   outPort ( 0x43, 0xB6, 1); // канал 2, операция 4, режим 3, формат 0
   iValue = iDivider & 0x0F;
   outPort ( 0x42, iValue, 1); // младший байт делителя
   outPort ( 0x42, ( iDivider >> 4), 1); // старший байт делителя
}
```

А теперь рассмотрим пример для получения случайного значения в установленном диапазоне (листинг 10.3).

Листинг 10.3. Получение случайного значения в диапазоне *от* 0 *до* 99

```
; выделяем место для переменной
Random_Value db ?
; сначала пишем процедуру для установки режима работы таймера
TimerInit proc near
mov AL, 10110111b   ; канал 2, операция 4, режим 3, формат 1
out 43h, AL  ; записываем значение в порт
mov AX, 11931; определяем делитель частоты
out 42h, AL  ; посылаем младший байт делителя
mov AL, AH   ; копируем значение
out 42h, AL  ; посылаем старший байт делителя
ret
TimerInit endp
; пишем процедуру получения случайного значения
GetRandomValue proc near
mov AL, 10000000b   ; канал 2, получить значение, биты 3-0 игнорируем
out 43h, AL  ; записываем значение в порт
in  AL, 42h  ; читаем младший байт значения счетчика
mov AH, AL   ; копируем его в AH
in  AL, 42h  ; читаем старший байт значения счетчика
```

```
call TimerInit      ; устанавливаем таймер заново
xchg AH, AL  ; меняем местами младший и старший байты
; сохраняем полученное значение в переменную
mov byte ptr Random_Value, AX
ret
GetRandomValue endp
```

Реализация генератора случайных чисел на C++ представлена в листинге 10.4.

Листинг 10.4. Получение случайного значения в C++

```
// пишем функцию установки таймера
void InitCount ( int iMaximum)
{
   int iValue = 0;
   int iDiv = 1193180 / iMaximum;
   outPort ( 0x43, 0xB7, 1); // канал 2, операция 4, режим 3, формат 1
   iValue = iDiv & 0x0F;
   outPort ( 0x42, iValue, 1); // младший байт делителя
   outPort ( 0x42, ( iDiv >> 4), 1); // старший байт делителя
}
// пишем функцию генерации случайного значения
int GetRandomValue ( int iMaximum)
{
   DWORD dwLSB = 0, dwMSB = 0;
   // канал 2, получить значение, биты 3-0 игнорируем
   outPort ( 0x43, 0x80, 1);
   // читаем младший байт значения счетчика
   inPort ( 0x42, &dwLSB, 1);
   // устанавливаем таймер заново
   InitCount ( iMaximum);
   // читаем старший байт значения счетчика
   inPort ( 0x42, &dwMSB, 1);
   return ( int) ( ( WORD) ( ( ( BYTE) ( dwLSB)) |
          ( ( ( WORD) ( ( BYTE) ( dwMSB))) << 8)));
}
```

На этом можно считать тему программирования системного таймера завершенной.