1. Работа с дисками на физическом уровне

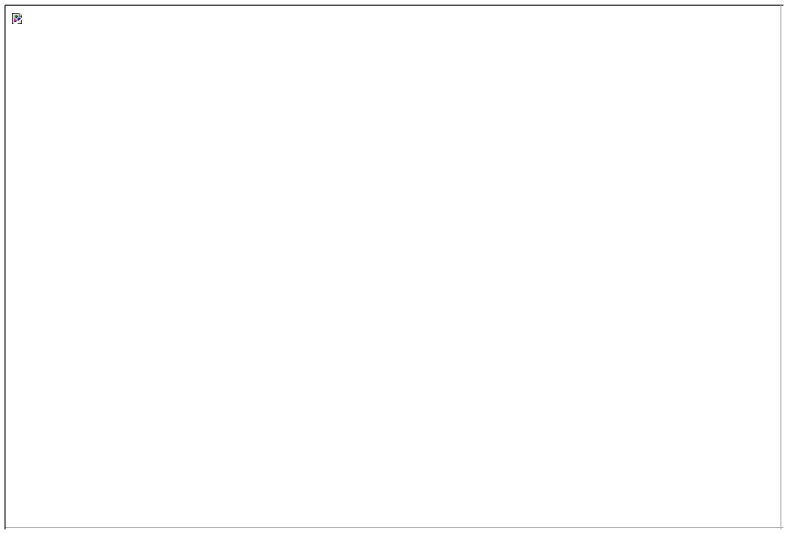
Начнем мы с того, что расскажем об аппаратном обеспечении дисковой подсистемы - о контроллерах и дисководах.

1.1. Дисководы и контроллеры

Первые персональные компьютеры фирмы IBM - IBM PC не имели жесткого диска ("винчестера", или, по отечественной терминологии, накопителя на жестком магнитном диске - НМД). Они были оборудованы двумя флоппи-дисками (накопителями на гибком магнитном диске - НГМД), которые и представляли собой дисковую подсистему. Отечественная персональная профессиональная ЭВМ (ППЭВМ) ЕС-1840 и некоторые модели ЕС-1841 также не имеют НМД. В таких компьютерах установлены, как правило, два дисковода для флоппи-дисков (дискет). Эти дисководы подключены к контроллеру - специальному устройству, находящемуся в корпусе персонального компьютера и выполняющему функции управления дисководами. Контроллер обычно выполнен в виде платы и вставлен в разъем общей шины, который находится на материнской плате (Motherboard) компьютера:

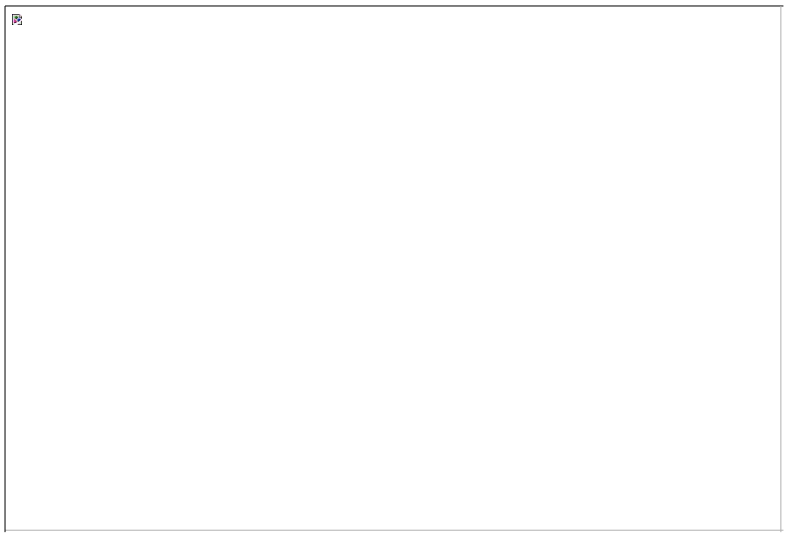
В оригинальном компьютере IBM PC и в отечественных ЕС1840/1841 используются флоппи-диски диаметром 5 дюймов.

Компьютер IBM XT и его ближайший отечественный аналог ЕС1841 может иметь один или два НГМД для дискет диаметром 5 дюймов и, как правило, один НМД емкостью 20 мегабайтов. Все дисководы подключаются к одному общему контроллеру, как это показано на рисунке:



Машины IBM AT и машины более высокого класса могут содержать несколько дисковых контроллеров, два флоппи-диска с различным диаметром (3 и 5 дюймов) и несколько жестких дисков. Впрочем, иногда обходятся одним флоппи-диском диаметром 5 или 3 дюйма. Если вы решили установить несколько дисковых контроллеров, необходимо позаботиться о том, чтобы эти контроллеры имели различные адреса на шине ввода/вывода компьютера. Это достигается правильной установкой соответствующих перемычек или переключателей, находящихся на плате контроллера, о чем подробно рассказано в документации на контроллер.

На рисунке показано, как могут быть подключены к машине AT несколько различных дисководов:



Покупая новый дисковод или контроллер, необходимо помнить о совместимости. Существует много типов контроллеров и дисководов, отличающихся используемым интерфейсом, способами записи информации и другими характеристиками. Эта книга не содержит конкретных рекомендаций по подбору дисководов и контроллеров, однако мы приведем несколько простых советов, которые помогут вам избежать некоторых неприятностей.

1. Покупайте дисковод вместе с тем контроллером, который для него предназначен. В этом случае совместимость дисковода и контроллера гарантируется.
2. Если вы покупаете дисковод для замены испорченного, выбирайте тот же тип и ту же фирму-изготовитель. Не надейтесь, что если дисковод имеет такую же емкость в мегабайтах, что и использовавшийся раньше, то он сможет работать в вашей системе.
3. Если дисковод должен работать в сети на сервере, убедитесь в том, что сетевая операционная система поддерживает данный тип дисковода и контроллера. Необходимая информация по этому вопросу находится в документации по сетевой операционной системе.

4. Не всегда удается заменить пятидюймовый НГМД на трехдюймовый. Убедитесь в том, что ваш контроллер может работать с трехдюймовым НГМД. Например, контроллер ППЭВМ ЕС-1841 не работает с такими дисководами.
5. Проверьте, достаточна ли мощность блока питания, встроенного в компьютер, для подключения новых дисководов, при необходимости используйте блок расширения.

Относительно недавно появились диски, расположенные непосредственно на плате контроллера, или, другими словами, совмещенные с контроллером (**Hardcard**). Эти диски имеют емкость несколько десятков мегабайтов, и их можно установить несколько штук.

Другая разновидность жестких дисков - сменные диски (**Cartridge**). В этих устройствах используется сменный магнитный носитель, иногда герметизированный и со встроенными элементами механического привода и встроенными магнитными головками. Емкость таких дисков составляет несколько десятков мегабайтов. Для сменных дисков используются специальные контроллеры и специальное программное обеспечение.

Пожалуй, самая интересная разновидность современных дисковых накопителей - оптические, или лазерные.

В настоящее время существует три типа оптических дисковых накопителей - **CD ROM, WORM** и стираемые диски.

Диски **CD ROM** (Compact-Disk, Read-Only Memory) - это диски, которые по своему формату и технологии записи информации напоминают компакт-диски для записи звука. Они имеют диаметр 120 миллиметров и могут содержать порядка 600 мегабайтов информации. Эта информация записывается один раз и впоследствии может только читаться, как из постоянного запоминающего устройства.

WORM-диски (Write Once, Read Many) предназначены для однократной записи и многократного считывания данных. Эти диски наилучшим образом подходят для архивного хранения информации, например, содержимого обширных баз данных.

Стираемые диски могут многократно использоваться для записи и чтения информации. Это самые дорогостоящие дисковые накопители.

Основной недостаток лазерных накопителей - относительно невысокое быстродействие по сравнению с традиционными накопителями на жестких дисках. Однако этот недостаток постепенно преодолевается и, по-видимому, оптические накопители скоро получат широкое распространение, особенно для работы с большими архивами и базами данных.

Для работы с оптическими накопителями используются специальные контроллеры и специальное программное обеспечение.

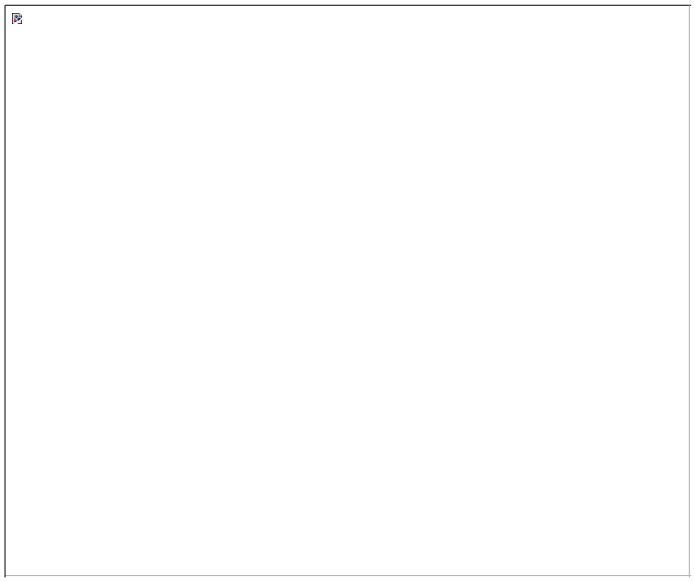
1.2. Сектора, головки, цилиндры...

Что же, собственно, представляет из себя диск?

Флоппи-диск - это круглая пластинка, покрытая с двух сторон магнитным материалом, напоминающим используемый в магнитных лентах для обычных бытовых магнитофонов, только отличающимся по некоторым характеристикам (например, по форме и ширине петли гистерезиза). Ближе к центру в диске находится маленькое отверстие, предназначенное для синхронизации:

Когда флоппи-диск вставляется в дисковод, с обеих сторон (сверху и снизу) к нему прижимаются магнитные головки. Они действительно прижимаются, зазора между головками и поверхностью флоппи-диска нет.

С помощью специального шагового двигателя головки могут перемещаться скачкообразно вдоль радиуса диска, как бы прочерчивая по поверхности диска концентрические окружности. Эти окружности называются дорожками, треками или цилиндрами - в литературе можно встретить различные названия:



Жесткий диск состоит из нескольких жестких круглых пластинок, покрытых магнитным материалом:

Эти пластинки вращаются с огромной скоростью (порядка 3600 оборотов в минуту) в герметичном корпусе. Как и для флоппи-диска, около каждой стороны пластинки располагается по одной магнитной головке, но эти головки не соприкасаются с поверхностью диска, а плавают на воздушной подушке в непосредственной близости от диска.

Подавая команды дисковому контроллеру, программа может перемещать блок головок вдоль радиуса диска, переходя таким образом от одного цилиндра к другому. Такие команды обычно выдаются не прикладной программой, а модулями BIOSBIOS, обслуживающими дисковый накопитель. Однако при необходимости программа может сама управлять положением блока

головок.

Перемещаясь вдоль окружности дорожки, магнитная головка может записывать или считывать информацию примерно так, как это происходит в бытовом магнитофоне. Запись производится по битам, при этом добавляется различная служебная информация и информация для контроля правильности данных.

Данные записываются не сплошным потоком, а блоками определенного размера. Эти блоки называются секторами. Сектор - это наименьший объем данных, который записывается или прочитывается контроллером. Для сектора выполняется контроль правильности записи или чтения. При записи сектора вычисляется контрольная сумма всех байтов, находящихся в секторе, и эта контрольная сумма записывается на диск в служебную область, находящуюся после сектора. При чтении эта контрольная сумма вычисляется заново и сравнивается со считанной из служебной области. При несовпадении контроллер сообщает программе об ошибке.

Дорожки нумеруются начиная от нулевой, головки тоже начиная от нулевой, а вот сектора - начиная с первого. Почему так было сделано - сказать трудно, но именно такая нумерация используется при работе с контроллером диска и функциями прерывания BIOSBIOS, обслуживающими дисковую подсистему.

Итак, подведем некоторые итоги.

- С точки зрения программы, работающей с контроллером НГМД или НМД, диск разбит на дорожки.
- Каждый диск обслуживают несколько головок, в зависимости от количества круглых пластинок, покрытых магнитным материалом, из которых состоит диск.
- Информация записывается и читается блоками, поэтому все дорожки как бы разбиты на сектора.
- В операциях чтения или записи на физическом уровне необходимо указывать номер дорожки (0,1,...), головки (0,1,...), номер сектора (1,2,...).

На самом деле для правильной работы с дисками на физическом уровне программа должна располагать существенно большей информацией о дисках, чем просто номера дорожек или головок. Например, она должна знать, сколько головок и сколько дорожек имеет то или иное дисковое устройство, сколько байтов содержится в одном секторе и многое другое.

Следущий раздел книги посвящен тому, как узнать конфигурацию дисковой подсистемы и основные параметры установленных дисковых накопителей.

1.3. Характеристики дисководов

Прежде чем начать работу с дисками на физическом уровне, необходимо выяснить конфигурацию дисковой подсистемы - сколько дисководов и какого типа подключено к компьютеру, сколько дорожек и головок имеется на каждом из дисководов и т.п. Способ, которым определяется конфигурация дисковой подсистемы, зависит от модели компьютера (PC, XT, AT), поэтому вначале займемся определением типа персонального компьютера.

ПЗУ BIOS BIOSсодержит по адресу **FFFF:FFFE** байт, значение которого можно использовать для идентификации типа компьютера:

FF	оригинальный IBM PC
FE	XT, Portable PC
FD	PCjr

FC	AT
FB	XT с памятью 640 К на материнской плате
F9	Convertible PC

Для компьютеров IBM PC и IBM XT конфигурация дисковой подсистемы определяется установкой переключателей на материнской плате, в частности, переключателями устанавливается количество подключенных к системе НГМД.

Машины IBM AT (и машины более высокого класса) имеют на материнской плате КМОП-память с малым энергопотреблением и питающуюся от аккумулятора (КМОП - это технология изготовления микросхем - КОМПЛЕМЕНТАРНАЯ пара МЕТАЛЛ-ОКИСЕЛ-ПОЛУПРОВОДНИК). В КМОП-памяти хранится информация о конфигурации дисковой подсистемы, при инициализации BIOS считывает эту информацию и записывает ее в свою внутреннюю область данных.

Для определения модели компьютера мы предлагаем следующую функцию:

```
/**
*.Name       pc_model
*
*.Title      Определить модель компьютера
*
*.Descr      Функция возвращает байт, идентифицирующий
*            модель персонального компьютера
*
*.Params     Нет
*
*.Return     Код модели персонального компьютера:
*
*               0xff - оригинальный PC;
*               0xfe - XT, Portable PC;
*               0xfd - PCjr;
*               0xfc - AT;
*               0xfb - XT с памятью 640K;
*               0xf9 - Convertible PC.
**/

#include <stdio.h>
#include <dos.h>
#include "sysp.h"

char unsigned pc_model(void) {

        char unsigned _far *modptr;

        modptr = FP_MAKE(0xf000,0xfffe);

        return *modptr;
}
```

Проанализировав значение, возвращаемое этой функцией, можно сделать предварительное заключение о конфигурации дисковой подсистемы компьютера. Если мы получили значения **0xff, 0xfd, 0xf9,** то наш компьютер не имеет НМД - это одна из разновидностей IBM PC. Значения **0xfe, 0xfb** могут соответствовать IBM XT и совместимым с ним машинам. Такие машины могут быть оборудованы НМД. И, наконец, значение **0xfc** соответствует IBM AT. Для этой машины конфигурация дисковой подсистемы должна определяться исходя из содержимого КМОП-памяти.

Следует заметить, что новые модели компьютеров могут иметь другие, не перечисленные выше, коды идентификации.

Прерывание BIOS **INT 11h** возвращает в регистре **AX** байт конфигурации системы, который можно использовать для определения количества НГМД и наличия НМД. Самый младший бит байта конфигурации - бит **0** - содержит признак наличия в системе НМД. Если этот бит установлен в **1**, то НМД присутствует в системе, иначе дисковая подсистема состоит только из накопителей на гибких магнитных дисках.

Биты **7** и **6** содержат информацию о количестве флоппи-дисков:

Содержимое битов 7 и 6	*Количество установленных флоппи-дисков*
00	1
01	2
10	3
11	4

Это прерывание лучше всего использовать для машин XT и PC. Для машин AT необходимо исследовать содержимое КМОП-памяти. Займемся этим.

КМОП-память не адресуема непосредственно из программы, как обычная оперативная память. Для работы с ней необходимо использовать команды ввода/вывода в порты с адресами **70h** и **71h.** Перед началом операции чтения/записи в порт **70h** надо записать адрес для КМОП-памяти (**0...3Fh**). Затем из порта **71h** можно прочитать содержимое требуемой ячейки КМОП-памяти или записать в этот порт байт, который будет записан в КМОП-память.

Приведем фрагмент программы, составленной на языке ассемблера, который считывает байт из КМОП-памяти с адресом **12h:**

```
mov al,12h
out 70h,al ; задаем адрес в КМОП-памяти
jmp $+2    ; небольшая задержка
in al,71h  ; записываем в AL считанное значение
```

Запись в КМОП-память выполняется аналогично.

При анализе конфигурации дисковой подсистемы для нас представляют наибольший интерес ячейки КМОП-памяти со следующими адресами:

```
14h - байт конфигурации
```

Биты **7, 6** этого байта имеют такое же значение, что и в младшем байте слова конфигурации, возвращаемого прерыванием BIOS **INT 11h** - они содержат информацию о количестве установленных дисководов для флоппи-дисков.

Значение бита **0**, равное нулю, говорит о том, что система не содержит НГМД.

```
10h - тип используемых флоппи-дисков
```

Младшая и старшая тетрады этого байта описывают соответственно второй и первый НГМД:

0000	дисковод не установлен;
0001	дисковод на 360К;
0010	дисковод на 1,2М.

0011	дисковод на 720K.
0100	дисковод на 1.44M.

```
12h - тип жестких дисков C: и D:
```

Этот байт разделен на две тетрады аналогично байту, описывающему НГМД. Однако в тетраде можно закодировать только **16** различных значений, а типов НМД значительно больше. Поэтому тип **15** используется специальным образом - если тип НМД в младшей тетраде (диск **C:**) равен **15**, то правильное значение типа находится в КМОП-памяти по адресу **19h**. Аналогично для диска **D:** этот тип можно взять из байта по адресу **1Ah** (если старшая тетрада байта с адресом **12h** равна **15**).

Если в вашем компьютере установлены диски с интерфейсом ESDI или SCSI или другим специализированным интерфейсом, то как правило, для работы с ними используется специальный "дисковый" BIOS. При этом в КМОП-памяти в ячейке **12h** для типа диска может быть указано нулевое значение, несмотря на то, что диск установлен. Прерывание BIOS **INT 11h** скажет вам, что в системе имеется НМД.

Если используется "дисковый" BIOS, то он сам инициализирует таблицу параметров диска и выполняет обработку дискового прерывания **INT 13h**. Так как MS-DOS для работы использует именно это прерывание, то не возникает никаких проблем, связанных с отсутствием типа диска в КМОП-памяти. Другие операционные системы, такие как XENIX и OS/2, могут использовать для работы с диском собственные драйверы. При установке они могут запрашивать информацию о типе установленного диска.

Если ваша машина содержит дисковый BIOS, то не исключено, что у вас будут проблемы при установке операционных систем XENIX и OS/2. В этом случае необходимо убедиться в том, что устанавливаемая операционная система содержит драйверы для работы с вашим типом диска.

Теперь мы готовы к тому, чтобы определить конфигурацию дисковой подсистемы - количество и типы используемых дисководов.

Приведем функцию, которая заполнит структуру типа **DISK_CONFIG**, описанную в файле **sysp.h,** информацией о конфигурации дисковой подсистемы.

Структура **DISK_CONFIG** содержит поля:

n_floppy	количество установленных в системе НГМД.
n_hard	количество установленных жестких НМД.
t_floppy1	тип первого НГМД.
t_floppy2	тип второго НГМД.
t_hard1	тип первого НМД.
t_hard2	тип второго НМД.

```
/**
*.Name      disk_cfg
*
*.Title     Определить конфигурацию дисковой подсистемы
*
*.Descr     Функция заполняет структуру, описывающую
*           конфигурацию дисковой подсистемы:
*
*              typedef struct _DISK_CONFIG_ {
*                 int  n_floppy;
*                 int  n_hard;
```

```
*                int  t_floppy1;
*                int  t_floppy2;
*                int  t_hard1;
*                int  t_hard2;
*             } DISK_CONFIG;
*
*
*.Params    Нет
*
*.Return    Ничего
**/

#include <stdio.h>
#include <dos.h>
#include "sysp.h"

void disk_cfg(DISK_CONFIG* cfg) {

        char unsigned _far *modptr;
        char unsigned pc_type;
        char cfg_byte;
        int  cfg_word;

        union REGS inregs, outregs;


// Определяем тип компьютера

        modptr = FP_MAKE(0xf000,0xfffe);
        pc_type = *modptr;

// В зависимости от типа компьютера выбираем
// способ определения конфигурации дисковой
// подсистемы

        switch (pc_type) {

                case 0xfc:

// Для IBM AT считываем конфигурацию дисковой
// подсистемы из КМОП-памяти

// Считываем байт конфигурации

                        outp(0x70, 0x14);
                        cfg_byte = inp(0x71);

// Определяем количество установленных флоппи-дисков

                        if((cfg_byte & 1) == 0) {

// Если младший бит байта конфигурации равен 0,
// флоппи-диски не установлены

                                cfg->n_floppy  = 0;
                                cfg->t_floppy1 = 0;
                                cfg->t_floppy2 = 0;

                        }
                        else {

// Определяем количество установленных
// флоппи-дисков

                                cfg->n_floppy = ((cfg_byte >> 6) & 3) + 1;

// Определяем типы флоппи-дисков

                                outp(0x70, 0x10);
                                cfg_byte = inp(0x71);

                                cfg->t_floppy2 = cfg_byte & 0xf;
```

```
                                cfg->t_floppy1 = (cfg_byte >> 4) & 0xf;

                        }

// Определяем конфигурацию жестких дисков

                        outp(0x70, 0x12);
                        cfg_byte = inp(0x71);

                        if(cfg_byte == 0) {

// Если обе тетрады равны нулю, система
// не содержит жестких дисков

                                cfg->n_hard  = 0;
                                cfg->t_hard1 = 0;
                                cfg->t_hard2 = 0;
                        }
                        else {

// Определяем тип первого диска - диска C:

                                if((cfg_byte & 0xf) != 0xf)
                                        cfg->t_hard1 = cfg_byte & 0xf;

                                else {
                                        outp(0x70, 0x19);
                                        cfg->t_hard1 = inp(0x71);
                                }

// Определяем тип второго диска - диска D:

                                if((cfg_byte & 0xf0) != 0xf0)
                                        cfg->t_hard2 = (cfg_byte >> 4) & 0xf;

                                else {
                                        outp(0x70, 0x1a);
                                        cfg->t_hard2 = inp(0x71);
                                }

                        }

// Вычисляем количество установленных
// в системе жестких дисков

                        cfg->n_hard = 0;
                        if(cfg->t_hard1 != 0) cfg->n_hard++;
                        if(cfg->t_hard2 != 0) cfg->n_hard++;

// Для некоторых совместимых с IBM AT машин невозможно
// определить тип диска, так как в КМОП-памяти для
// типа диска установлено значение 0, несмотря на то,
// что диск установлен (например машина Bondwell,
// модель B-300). В таких случаях можно определить
// наличие жесткого диска, используя слово
// конфигурации, возвращаемое прерыванием INT 11h.

                        if(cfg->n_hard == 0) {

                                int86(0x11, &inregs, &outregs);
                                cfg_word = outregs.x.ax;

// Определяем наличие жесткого диска

                                if((cfg_word & 1) != 0) {

                                        cfg->n_hard =  1;

// Считаем, что тип используемого жесткого
// диска неопределен

                                        cfg->t_hard1 = 0;
```

```
                                        cfg->t_hard2 = 0;

                                }
                        }

                break;

                default:

// Для остальных типов компьютеров вызываем
// прерывание INT 11h, используем возвращаемый
// этим прерыванием байт конфигурации

                        int86(0x11, &inregs, &outregs);
                        cfg_word = outregs.x.ax;

// Определяем количество установленных
// флоппи-дисков

                        cfg->n_floppy = ((cfg_word >> 6) & 3) + 1;

// Считаем, что тип используемого флоппи-диска
// неопределен

                        cfg->t_floppy1 = 0;
                        cfg->t_floppy2 = 0;

// Определяем наличие жесткого диска

                        if((cfg_word & 1) != 0) {

                                cfg->n_hard =  1;

// Считаем, что тип используемого жесткого
// диска неопределен

                                cfg->t_hard1 = 0;
                                cfg->t_hard2 = 0;

                        }

                break;

        }

}
```

Пользуясь приведенной выше функцией мы всегда сможем определить количество дисководов для флоппи-дисков и жестких дисков, но не всегда сможем определить их тип. Это само по себе не страшно, так как для работы с дисками на физическом уровне нам надо знать не столько тип диска, сколько другие его характеристики, такие как количество головок, секторов и др. Эти характеристики можно определить из таблиц параметров для дискет и жестких дисков, заполняемых модулями BIOS процессе инициализации системы.

Приведем сокращенную таблицу параметров для стандартных типов жестких дисков, возвращаемых функцией **disk_cfg**. Информация, которая содержится в этой таблице, используется BIOS процессе инициализации, когда модули BIOS анализируют содержимое КМОП-памяти.

Тип	*Количество цилиндров*	*Количество головок*	*Емкость диска в байтах*
1	306	4	10.653.696
2	615	4	21.411.840
3	615	6	32.117.760

4	940	8	65.454.080
5	940	6	49.090.560
6	615	4	21.411.840
7	462	8	32.169.984
8	733	5	31.900.160
9	900	15	117.504.000
10	820	3	21.411.840
11	855	5	37.209.600
12	855	7	52.093.440
13	306	8	21.307.392
14	733	7	44.660.224
15	0	0	0
16	612	4	21.307.392
17	977	5	42.519.040
18	977	7	59.526.656
19	1024	7	62.390.272
20	733	5	31.900.160
21	733	7	44.660.224
22	733	5	31.900.160
23	306	4	10.653.696
24	977	5	42.519.040
25	1024	9	80.216.064
26	1224	7	74.575.872
27	1224	11	117.190.656
28	1224	15	159.805.440
29	1024	8	71.303.168
30	1024	11	98.041.856
31	918	11	87.892.992
32	925	9	72.460.800
33	1024	10	89.128.960
34	1024	12	106.954.752
35	1024	13	115.867.648
36	1024	14	124.780.544
37	1024	2	17.825.792
38	1024	16	142.606.336
39	918	15	119.854.080
40	820	6	42.823.680

Для всех приведенных в таблице типов дисков на цилиндре (на дорожке) располагается **17** секторов.

Стандартная машина IBM XT комплектуется обычно НМД с типом 1, тип 2 используется

стандартной IBM AT. Остальные типы НМД поддерживаются не всеми версиями BIOS, например, типы 16...23 поддерживаются BIOS только тех версий, которые были изготовлены не позднее 15/11/85.

Наиболее широко распространены флоппи-диски емкостью 360K, 1.2M, 720K, 1.44M. Их параметры приведены в следующей таблице:

Тип	*Емкость, Кбайтов*	*Диаметр, дюймы*	*Количество секторов на одну дорожку*	*Количество цилиндров*
1	360	5	9	40
2	1200	5	15	80
3	720	3	9	40
4	1440	3	18	80

Тип дискеты в приведенной таблице соответствует возвращаемому функцией **disk_cfg.**

Анализируя содержимое КМОП-памяти в машинах AT или установку переключателей конфигурации на материнской плате в машинах PC и XT, BIOS процессе инициализации создает таблицу параметров дискеты **DPT** (Diskette Parameter Table), а также одну или две таблицы параметров жесткого диска **HDPT** (Hard Disk Parameter Table). Если имеется специальный дисковый BIOS, то он сам создает таблицы **HDPT**.

Таблица параметров дискеты **DPT** имеет длину 10 байт, ее адрес располагается в области данных **BIOS** по адресу **0000:0078**, что соответствует вектору прерывания **INT 1Eh**. Таблица содержит параметры, важные для работы дисковода:

(0) 1	srt_hut	Биты 0...3 - SRT (Step Rate Time) - задержка для переключения головок, лежит в пределах 1-16 мс и задается с интервалом 1 мс (0Fh - 1mc, 0Eh - 2 mc, 0Dh - 3 mc, ...); биты 4...7 - задержка разгрузки головки, лежит в пределах 16-240 мс и задается с интервалом 16 мс (1 - 16 mc, 2 - 32 mc, ..., 0Fh - 240 mc).
(+1) 1	dma_hlt	Бит 0 - значение этого бита, равное 1, говорит о том, что используется прямой доступ к памяти (DMA); биты 2...7 - время загрузки головок HLT - интервал между сигналом загрузки головок и началом операции чтение/запись, лежит в пределах 2-254 мс и задается с интервалом 2 мс (1 - 2 mc, 2 - 4 mc, ..., 0FFh - 254 mc).
(+2) 1	motor_w	Задержка перед выключением двигателя.
(+3) 1	sec_size	Код размера сектора в байтах (0 - 128 байтов, 1 - 256, 2 - 512, 3 - 1024).
(+4) 1	eot	Номер последнего сектора на дорожке
(+5) 1	gap_rw	Длина межсекторного промежутка для чтения/записи.
(+6) 1	dtl	Максимальная длина передаваемых данных, используется когда не задана длина сектора.

(+7) 1	gap_f	Длина межсекторного промежутка для операции форматирования.
(+8) 1	fill_char	Байт-заполнитель для форматирования (обычно используется F6h).
(+9) 1	hst	Время установки головки в миллисекундах.
(+10) 1	mot_start	Время запуска двигателя в 1/8 долях секунды.

Все времена в таблице зависят от частоты тактового генератора контроллера НГМД, приведенные значения соответствуют частоте 8 МГц.

Для удобства работы с таблицей параметров дискеты файл **sysp.h** содержит определение типа **DPT**:

```
#pragma pack(1)

typedef struct _DPT_ {
        unsigned char srt_hut;
        unsigned char dma_hlt;
        unsigned char motor_w;
        unsigned char sec_size;
        unsigned char eot;
        unsigned char gap_rw;
        unsigned char dtl;
        unsigned char gap_f;
        unsigned char fill_char;
        unsigned char hst;
        unsigned char mot_start;
} DPT;

#pragma pack()
```

Адреса таблиц параметров жестких дисков **HDPT** расположены по адресам, соответствующим векторам прерываний **INT 41h** (для первого физического диска) и **INT 46h** (для второго физического диска). Эти таблицы имеют следующий формат:

(0) 2	max_cyl	Максимальное количество цилиндров на диске.
(+2) 1	max_head	Максимальное количество магнитных головок.
(+3) 2	srwcc	Начальный цилиндр для предварительной записи (Starting reduced-write current cylinder).
(+5) 2	swpc	Начальный цилиндр для предварительной компенсации при записи (Starting write precompensation cylinder).
(+7) 1	max_ecc	Максимальная длина блока коррекции ошибок ECC (Maximum ECC data burst length).
(+8) 1	dstopt	Опции устройства: бит 7 - запрет восстановления; бит 6 - запрет восстановления по блоку коррекции ошибок ECC (Error Correction Code); биты 2-0 - опции устройства.
(+9) 1	st_del	Стандартная величина задержки.
(+10) 1	fm_del	Величина задержки для форматирования диска.

(+11) 1	chk_del	Величина задержки для проверки диска.
(+12) 4	reserve	Зарезервировано.

Файл **sysp.h** содержит соответствующее определение типа **HDPT**:

```
#pragma pack(1)

typedef struct _HDPT_ {
        unsigned max_cyl;
        unsigned char max_head;
        unsigned srwcc;
        unsigned swpc;
        unsigned char max_ecc;
        unsigned char dstopt;
        unsigned char st_del;
        unsigned char fm_del;
        unsigned char chk_del;
        char reserve[4];
} HDPT;

#pragma pack()
```

Наиболее полезная информация, которую можно извлечь из таблицы параметров дискеты - это код размера сектора. Если вам когда-либо понадобится работать с нестандартным размером сектора (512 байтов), вам не обойтись без этой таблицы.

Таблица параметров жесткого диска содержит такие важнейшие значения, как максимальное количество цилиндров и максимальное количество головок. Если вам не удалось определить тип диска, то таблица **HDPT** - единственное надежное место, откуда можно получить информацию о цилиндрах и головках.

Для удобства использования таблиц параметров дискет и дисков мы подготовили следующие функции:

```
/**
*.Name      get_dpt
*
*.Title     Вычислить адрес таблицы параметров дискеты
*
*.Descr     Функция возвращает указатель на таблицу
*           параметров дискеты
*
*.Params    Нет
*
*.Return    Указатель на таблицу параметров дискеты DPT
**/

#include <stdio.h>
#include <dos.h>
#include "sysp.h"

DPT _far *get_dpt(void) {
void _far * _far *ptr;

        ptr = (void _far * _far *)FP_MAKE(0x0,0x78);
        return(*ptr);
}
/**
*.Name      get_hdp1
*
*.Title     Вычислить адрес таблицы параметров диска 1
*
*.Descr     Функция возвращает указатель на таблицу
*           параметров диска 1
*
```

```
*.Params    Нет
*
*.Return    Указатель на таблицу параметров диска 1 HDPT
**/

#include <stdio.h>
#include <dos.h>
#include "sysp.h"

HDPT _far *get_hdp1(void) {
void _far * _far *ptr;

        ptr = (void _far * _far *)FP_MAKE(0x0,0x104);
        return(*ptr);

}
/**
*.Name      get_hdp2
*
*.Title     Вычислить адрес таблицы параметров диска 2
*
*.Descr     Функция возвращает указатель на таблицу
*           параметров диска 2
*
*.Params    Нет
*
*.Return    Указатель на таблицу параметров диска 2 HDPT
**/

#include <stdio.h>
#include <dos.h>
#include "sysp.h"

HDPT _far *get_hdp2(void) {
void _far * _far *ptr;

        ptr = (void _far * _far *)FP_MAKE(0x0,0x118);
        return(*ptr);

}
```

В качестве примера приведем программу, которая определяет конфигурацию дисковой подсистемы и отображает основные характеристики используемых дисководов. Программа обращается к таблицам параметров НГМД и НМД:

```
#include <stdio.h>
#include <dos.h>
#include "sysp.h"

void main(void);
void main(void) {

  DISK_CONFIG cfg;
  DPT  _far *dpt_ptr;
  HDPT _far *hdpt1_ptr;
  HDPT _far *hdpt2_ptr;

  printf("\n"
                        "\nКонфигурация дисковой подсистемы"
                        "\n  (C)Фролов А., 1991"
                        "\n");

// Определяем конфигурацию дисковой подсистемы

  disk_cfg(&cfg);

  printf("\nУстановлено:"
                        "\n   Флоппи-дисков: %d"
                        "\n   Дисков:        %d",
                        cfg.n_floppy,
```

```
                        cfg.n_hard);

  printf("\nТипы флоппи-дисков:  A: - %d, B: - %d"
                        "\nТипы дисков:         C: - %d, D: - %d",
                        cfg.t_floppy1, cfg.t_floppy2,
                        cfg.t_hard1,   cfg.t_hard2);

// Получаем адрес таблицы параметров дискеты

  dpt_ptr = get_dpt();

  printf("\n"
                        "\nКод размера сектора дискеты:                    %d"
                        "\nЗаполняющий символ для форматирования дискеты: %2.2X",
                        dpt_ptr->sec_size,
                        dpt_ptr->fill_char);

// Получаем адреса первой и второй таблицы
// параметров жесткого диска

  hdpt1_ptr = get_hdp1();
  hdpt2_ptr = get_hdp2();

  printf("\n"
                        "\nПараметры первого диска:"
                        "\n   Количество цилиндров:     %d"
                        "\n   Количество головок:       %d"
                        "\n"
                        "\nПараметры второго диска:"
                        "\n   Количество цилиндров:     %d"
                        "\n   Количество головок:       %d",
                        hdpt1_ptr->max_cyl,
                        hdpt1_ptr->max_head,
                        hdpt2_ptr->max_cyl,
                        hdpt2_ptr->max_head);

}
```

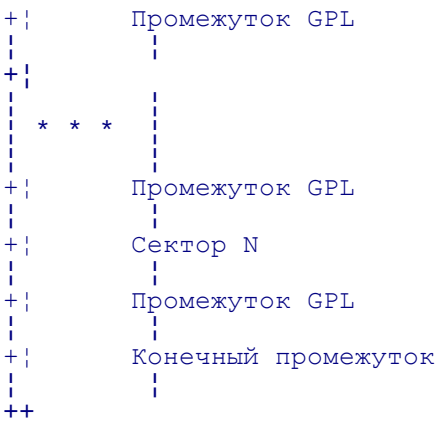
1.4. Программирование контроллера НГМД

Большинство дисковых операций можно выполнить на уровне функций BIOS. Это самый простой и надежный способ работы с диском на физическом уровне. Однако в отдельных случаях вам может потребоваться непосредственный доступ к контроллеру НГМД - например, если вы разрабатываете систему защиты данных от копирования.

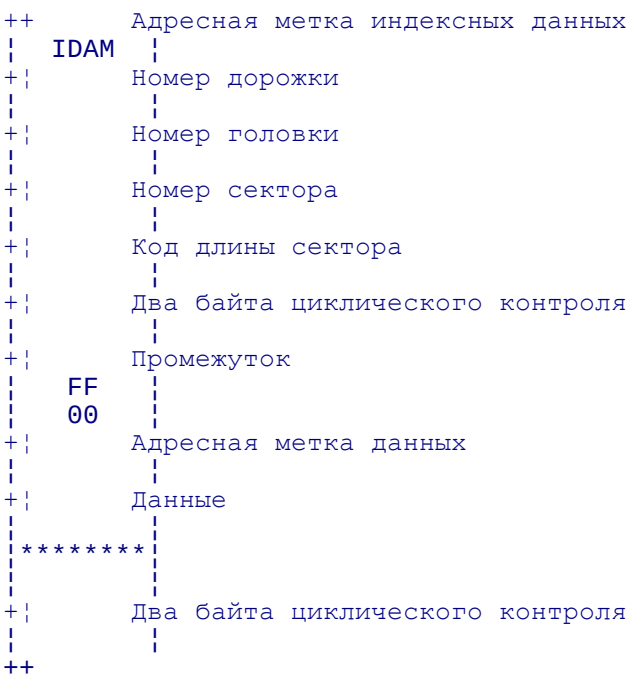
Информация, приведенная в этом разделе, ориентирована прежде всего не на выполнение операций чтения/записи (которые лучше выполнять с помощью функций BIOS), а на управление контроллером и получение состояния контроллера. Именно эти возможности требуются для организации защиты данных от копирования.

Для лучшего понимания работы контроллера мы приведем схему расположения зон данных на дорожке флоппи-диска:

```
++      Прединдексный синхронизирующий промежуток
¦   FF   ¦
¦   00   ¦
+¦      Индексная адресная метка
¦  IAM   ¦
+¦      Промежуток 1
¦        ¦
+¦      Сектор 1
¦        ¦
+¦      Промежуток GPL
¦        ¦
+¦      Сектор 2
¦        ¦
```



Каждый сектор на дорожке состоит из областей индексных данных и данных. Сектора разделены промежутком **GPL**, в конце дорожки располагается конечный промежуток, его размер зависит от скорости вращения диска, длин секторов и других промежутков. Область индексных данных содержит информацию о номере дорожки, головки, сектора, код длины сектора. Область данных содержит сами данные. Приведем формат сектора:



Программа обращается к контроллеру для выполнения различных операций с помощью команд ввода/вывода. Для машин IBM PC и XT используются три порта с адресами **3F2h, 3F4h** и **3F5h**. В машинах класса AT дополнительно используются два порта с адресами **3F6h** и **3F7h.**

Порт **3F2h** работает только на запись, это порт вывода. С его помощью можно выбирать для работы один из дисководов (одновременно можно работать только с одним дисководом), сбрасывать контроллер в исходное состояние, разрешать или запрещать прерывания от контроллера и работу схем прямого доступа к памяти, включать или выключать двигатели дисководов.

Назначение отдельных битов этого порта:

0-1	Выбор дисковода. Машины AT не используют бит 1, так как в этих машинах только два НГМД.

2	0 - сброс контроллера; 1 - разрешение работы контроллера.
3	1 - разрешение прерываний и прямого доступа к памяти.
4-7	Значение 1 в каждом разряде вызывает включение соответствующего двигателя дисковода. Для машин AT биты 6-7 не используются.

Порт **3F4h** предназначен только для чтения. С его помощью можно получить байт основного состояния контроллера. Назначение битов:

0-3	Значение 1 говорит о том, что соответствующий дисковод занят, он выполняет операцию поиска. Для машины AT биты 2-3 не используются.
4	Контроллер занят выполнением команды чтения или записи.
5	0 - используется режим прямого доступа к памяти; 1 - режим прямого доступа к памяти не используется.
6	Направление передачи данных: 0 - от процессора к контроллеру; 1 - от контроллера к процессору.
7	Запрос на передачу данных - контроллер готов к записи или чтению данных.

Порт **3F5h** предназначен для записи или чтения данных. Он используется при всех операциях контроллера.

Выполнение любой операции начинается с того, что программа посылает в этот порт байт кода операции, за которым следует один или несколько байтов параметров. Количество байтов параметров и их назначение зависит от кода операции (т.е. от первого байта). После выполнения операции программа считывает несколько байтов результата для анализа правильности выполнения операции.

Порт **3F7h** работает на запись и чтение, он используется только в машинах AT.

При записи биты **0-1** определяют скорость передачи данных:

00	500 Кбайтов/с (высокая плотность HD);
01	300 Кбайтов/с (двойная плотность DD);
10	250 Кбайтов/с (одинарная плотность SD);
11	зарезервировано.

Приведем назначение отдельных битов порта 3F7h для чтения:

0	1 - выбран дисковод 0
1	1 - выбран дисковод 1
2-5	Выбраны головки, бит 2 соответствует головке 0, бит 3 - головке 1 и т.д.
6	Переключатель записи.
7	1 - признак замены дискеты.

Контроллер НГМД может выполнять 15 операций, или команд. Выполнение команды разделяется на три фазы - командная фаза, фаза выполнения, фаза результата. В командной фазе программа должна передать контроллеру всю информацию, необходимую для выполнения команды. В фазе выполнения команда выполняется, и в фазе результата программа получает от контроллера информацию о состоянии контроллера.

Информация, необходимая для выполнения команды, передается контроллеру через порт данных **3F5h**. В соответствии с форматом команды программа должна последовательно вывести в этот порт код команды и все параметры.

Прежде чем программа начнет командную фазу, она должна убедиться в том, что контроллер завершил выполнение предыдущей операции и готов к приему команды. Для этого программа должна считать байт основного состояния контроллера из порта с адресом **3F4h** и проверить биты **6** и **7**. Бит **6** должен быть установлен в **0**. Это означает, что данные будут передаваться от процессора к контроллеру. Бит **7** должен быть установлен в **1** - это готовность контроллера к приему команды.

Фаза выполнения начинается после установки битов **6** и **7** байта основного состояния в **1**. После завершения выполнения команды контроллер формирует сигнал запроса прерывания.

В фазе результата процессор считывает состояние контроллера. Это состояние хранится в нескольких внутренних регистрах контроллера:

RS - регистр основного состояния;

ST0, ST1, ST2, ST3 - регистры дополнительного состояния.

Регистр основного состояния доступен через порт **3F4h**, содержимое остальных регистров процессор считывает после выполнения контроллером команды через порт данных **3F5h.**

Приведем форматы для всех команд контроллера НГМД.

```
Команда Байты команды

Чтение данных
                ++
                ¦MT ¦MFM¦SK ¦ 0 ¦ 0 ¦ 1 ¦ 1 ¦ 0 ¦
                ++++++++¦
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦HDS¦DS1¦DS0¦
                ++

Чтение удаленных данных
                ++
         ¦      MT ¦MFM¦SK ¦ 0 ¦ 1 ¦ 1 ¦ 0 ¦ 0 ¦
                ++++++++¦
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦HDS¦DS1¦DS0¦
                ++

Запись данных
                ++
                ¦MT ¦MFM¦ 0 ¦ 0 ¦ 0 ¦ 1 ¦ 0 ¦ 0 ¦
                ++++++++¦
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦HDS¦DS1¦DS0¦
                ++

Запись удаленных данных
                ++
                ¦MT ¦MFM¦ 0 ¦ 0 ¦ 1 ¦ 0 ¦ 0 ¦ 1 ¦
                ++++++++¦
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦HDS¦DS1¦DS0¦
                ++

Чтение данных с дорожки
```

```
                ++
                ¦MT ¦MFM¦SK ¦ 0 ¦ 0 ¦ 0 ¦ 1 ¦ 0 ¦
                +++++++¦
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦HDS¦DS1¦DS0¦
                ++
```

Сканирование до "равно"

```
                ++
                ¦MT ¦MFM¦SK ¦ 1 ¦ 0 ¦ 0 ¦ 0 ¦ 1 ¦
                +++++++¦
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦HDS¦DS1¦DS0¦
                ++
```

Сканирование до "меньше" или "равно"

```
                ++
                ¦MT ¦MFM¦SK ¦ 1 ¦ 1 ¦ 0 ¦ 0 ¦ 1 ¦
                +++++++¦
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦HDS¦DS1¦DS0¦
                ++
```

Сканирование до "больше" или "равно"

```
                ++
                ¦MT ¦MFM¦SK ¦ 1 ¦ 1 ¦ 1 ¦ 0 ¦ 1 ¦
                +++++++¦
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦HDS¦DS1¦DS0¦
                ++
```

Форматирование дорожки

```
                ++
                ¦ 0 ¦MFM¦ 0 ¦ 0 ¦ 1 ¦ 1 ¦ 0 ¦ 1 ¦
                +++++++¦
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦HDS¦DS1¦DS0¦
                ++
```

Считывание индексных данных

```
                ++
                ¦ 0 ¦MFM¦ 0 ¦ 0 ¦ 1 ¦ 0 ¦ 1 ¦ 1 ¦
                +++++++¦
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦HDS¦DS1¦DS0¦
                ++
```

Инициализация

```
                ++
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 1 ¦ 1 ¦ 1 ¦
                +++++++¦
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 1 ¦ 0 ¦
                ++
```

Чтение состояния прерывания

```
                ++
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 1 ¦ 0 ¦ 0 ¦ 0 ¦
                ++
```

Определить параметры

```
                ++
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 1 ¦ 1 ¦
                ++
```

Чтение состояния накопителя

```
                ++
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 1 ¦ 0 ¦ 0 ¦
                +++++++¦
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦HDS¦DS1¦DS0¦
                ++
```

Поиск

```
                ++
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 1 ¦ 1 ¦ 1 ¦ 1 ¦
                +++++++¦
                ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦ 0 ¦HDS¦DS1¦DS0¦
                ++
```

Первые несколько команд имеют одинаковый формат параметров и байтов результата.

Приведем байты параметров, которые должны следовать за командами и байты результата, которые процессор должен считать после выполнения команды.

Команда	*Байты параметров*	*Байты результата*
Чтение данных	C, H, R, N, EOT,EOT, GPL, DTL	ST0, ST1, ST2,C, H, R, N
Чтение удаленных данных		
Запись данных		
Запись удаленных данных		
Чтение данных с дорожки		
Сканирование до "равно"		
Сканирование до "меньше" или "равно"		
Сканирование до "больше" или "равно"		
Форматирование дорожки	N, SC, GPL, D	ST0, ST1, ST2,C, H, R, N
Чтение индексных данных	отсутствуют	ST0, ST1, ST2,C, H, R, N
Инициализация	отсутствуют	отсутствуют
Чтение состояния прерывания	отсутствуют	ST0, PCN
Определить параметры	1 байт: мл. тетрада - HUT ст. тетрада - SRT 2 байт: бит 0 - ND биты 1-7 - HLT	отсутствуют
Чтение состояния накопителя	отсутствуют	ST3
Поиск	C	отсутствуют

После выполнения команды центральный процессор должен получить от контроллера байты результата. Среди них - содержимое внутренних регистров состояния контроллера **ST0, ST1, ST2, ST3**. Опишем назначение отдельных битов этих регистров.

Формат регистра ST0:

Биты	*Название*	*Назначение*

1, 0	US1, US2	Эти биты содержат код накопителя при прерывании
2	HD	Номер головки.
3	NC	Накопитель не готов, устанавливается,если накопитель не готов выполнить команду чтения или записи.
4	EC	Сбой оборудования
5	SE	Завершена команда "Поиск"
7, 6	I, C	Код прерывания: 00 - нормальное завершение; 01 - аварийное завершение; 10 - некорректная команда; 11 - нет готовности дисковода.

Формат регистра ST1:

Биты	*Название*	*Назначение*
0	MA	Пропуск адресной метки. Этот бит устанавливается в 1, если контроллер не может найти адресную метку
1	NN	Защита записи, устанавливается , если при выполнении операции контроллер получает от дисковода сигнал защиты записи.
2	ND	Не найден сектор.
3	-	Зарезервирован
4	OR	Переполнение, процессор не успевает выполнять обмен данными с контроллером
5	DE	Ошибка в данных при проверке контрольной суммы
6	-	Зарезервирован.
7	EN	Несуществующий сектор, устанавливается, когда контроллер пытается прочесть сектор с адресом, большим существующего.

Формат регистра ST2:

Биты	*Название*	*Назначение*
0	MD	Пропущен адресный маркер в поле данных.
1	BC	Нечитающаяся дорожка.
2	SN	Ошибка сканирования. Устанавливается, если при выполнении команды сканирования контроллер не может найти требуемую дорожку.
3	SH	Сканирование выполнено, дорожка найдена.
4	WC	Ошибка адреса дорожки.
5	DD	Ошибка в поле данных.

6	CM	Во время операции чтения или сканирования не обнаружен сектор с маркером удаленных данных.
7	-	Зарезервирован.

Формат регистра ST3:

Биты	*Название*	*Назначение*
1, 0	US1, US2	Код выбранного дисковода.
2	HD	Номер выбранной головки.
3	TS	Используется режим двухсторонней записи.
4	T0	Головка установлена на дорожку 0.
5	RDY	Дисковод готов к работе.
6	WP	Защита записи на диске.
7	FT	Неисправность дисковода.

Дополнительно перед выполнением операции и после ее завершения надо проанализировать содержимое описанного выше регистра основного состояния контроллера RS.

В форматах команд и таблицах используются следующие обозначения:

MT	двухсторонняя операция
MFM	двойная/одинарная плотность записи
SK	пропуск удаленных данных
HDS	номер головки для двухстороннего накопителя
DS1, DS0	номер выбираемого накопителя
C	номер цилиндра
H	номер головки для двухстороннего накопителя
R	номер сектора
N	число байтов в секторе
EOT	номер последнего сектора на дорожке
GPL	размер промежутка
DTL	число считываемых/записываемых байтов
SC	число секторов в цилиндре
D	данные
PCN	номер цилиндра после выполнения команды чтения состояния прерывания
SRT	время шага, мс
HUT	время разгрузки головки
HLT	время загрузки головки
ND	режим прерывания
NCN	номер цилиндра после поиска

Команда "Определить параметры" задает времена задержки для трех внутренних таймеров контроллера. Первый байт параметров состоит из двух полей - **SRT** и **HUT**. Поле **SRT** задает временной интервал между шаговыми импульсами двигателя перемещения головки. Это поле имеет ширину 4 бита. Поле **HUT** определяет время разгрузки головки и тоже имеет ширину 4 бита.

Второй байт параметров состоит из полей **HLT** и **ND**. Поле **HLT** имеет ширину 7 битов и определяет время загрузки головки. Бит **ND** определяет использование канала прямого доступа - если этот бит установлен в 0, то ПДП используется, иначе обмен данными идет через центральный процессор.

Параметры для команды "Определить параметры" лучше всего взять из таблицы параметров дискеты, заполняющейся BIOS во время инициализации системы. Конечно, если вам нужны нестандартные параметры, вы можете попробовать использовать свои, ориентируясь на значения из таблицы параметров дискеты.

Команда "Инициализация" может выполняться одновременно для всех накопителей. По этой команде головки перемещаются на нулевую дорожку.

Команда "Поиск" используется для установки головки на нужную дорожку. Поиск может выполняться одновременно для нескольких накопителей.

Команда "Чтение состояния прерывания" может вырабатываться после завершения других команд для выяснения состояния контроллера после прерывания. Эту команду удобно использовать после команд "Поиск" или "Инициализация".

После поступления ***команды "Чтение данных"*** загружается головка, контроллер считывает метки адреса идентификатора **ID** и поля **ID**. Контроллер последовательно считывает номера секторов, и как только считанный номер совпадет с требуемым, считывает данные сектора байт за байтом и передает их либо центральному процессору, либо каналу прямого доступа к памяти. При передаче данных контроллер должен обслуживаться каждые **27** мкс в режиме одинарной плотности и **13** мкс в режиме двойной плотности, иначе в регистре состояния **ST3** устанавливается флаг переполнения OR**.**

Если контроллер не может найти нужный сектор, то в регистре **ST1** устанавливается флаг отсутствия данных **ND**. При ошибке считывания данных, обнаруженной схемами избыточного циклического контроля **CRC**, устанавливается флаг ошибки данных **DE.**

При считывании адресной метки удаленных данных в регистре **ST2** и сброшенном в **0** бите **SK** команды флаг **CM** устанавливается в **1**, читаются все данные из этого сектора, затем выполнение команды прекращается.

Поле команды **MT** позволяет задать выполнение многодорожечной операции, при которой контроллер считывает данные с обеих сторон дискеты. Поле **MFM** определяет плотность обрабатываемой информации: значение **0** соответствует одинарной плотности, **1** - двойной.

Если поле команды **N** содержит **0**, то поле **DTL** определяет объем передаваемых данных. Если поле **N** содержит отличное от нуля значение, поле **DTL** игнорируется и должно содержать значение **0FFh**.

Выполнение ***команды "Запись"*** аналогично. В режиме записи обмен данными процессора с контроллером должен происходить каждые **31** мкс в режиме одинарной плотности и каждые **15** мкс в режиме двойной плотности.

По команде "Запись удаленных данных" в начале поля данных записывается адресная метка

удаленных данных вместо обычной адресной метки данных.

По команде "Чтение данных дорожки" считываются все поля данных с каждого сектора дорожки как непрерывные блоки данных. С помощью этой команды можно производить многодорожечные операции, пропуски.

Команда "Чтение индексных данных" позволяет определить положение головки.

Команда "Форматирование дорожки" форматирует всю дорожку - на нее записываются интервалы, адресные метки, поля индексных данных и поля данных. Вам не обязательно располагать сектора в порядке увеличения номеров, т.к. при форматировании контроллер запрашивает параметры **C, H, R, N.**

Группа команд "Сканирование" позволяет сравнивать данные, поступающие от контроллера и от центрального процессора. Контроллер выполняет побайтное сравнение и ищет сектор, удовлетворяющий заданному условию. При выполнении условия сканирования в регистре состояния **ST2** устанавливается флаг **SH**, в противном случае - флаг **SN**.

Как пользоваться всеми этими командами?

Выполнив сброс контроллера, вам надо его проинициализировать, задав все рабочие параметры. Затем можно выдавать контроллеру команды, каждый раз проверяя регистр его основного состояния **ST** и анализируя байты результата **ST0...ST3**. Можно предложить следующую последовательность действий:

- Сброс контроллера выдачей в порт **3F2h** байта с битом **2**, установленным в **0**.
- Разрешение работы контроллера выдачей в этот же порт байта с битом **2**, установленным в **1**.
- Выдача контроллеру команды "Инициализация".
- Выдача контроллеру команды "Определить параметры".
- Включить двигатель и выждать примерно 0,5 с (время разгона двигателя).
- Установить головки в нужное положение командой "Поиск".
- Проверить результаты установки командой "Чтение состояния прерывания".
- Для машины AT установить нужную скорость передачи данных, выдав в порт **3F7h** байт с соответствующим значением: **0** для дискет с высокой плотностью записи (HD), **1** для двойной плотности (DD) и **2** для одинарной (SD).
- Если установка головок произведена правильно, можно выдавать команды чтения/записи данных. Перед этим надо правильно запрограммировать контроллер прямого доступа к памяти, если вы собираетесь использовать режим ПДП.

Программирование контроллера прямого доступа к памяти будет подробно описано во втором томе книги, сейчас мы приведем только основные сведения, необходимые для того, чтобы разобраться в программе, демонстрирующей использование команд контроллера НГМД.

Контроллер прямого доступа к памяти (КПДП) имеет несколько каналов и для машин AT состоит из двух микросхем Intel 8237A. Контроллер НГМД использует канал **2.**

Перед началом инициализации КПДП программа должна послать в порты **0Bh** и **0Ch** код операции, которая будет выполняться КПДП - **46h** для операции чтения и **4Ah** для операции записи.

В процессе инициализации программа должна сообщить КПДП адрес буфера, куда ему следует поместить данные или откуда надо взять данные, и длину передаваемых данных в байтах.

Адрес необходимо представить в виде номера страницы и смещения. Для КПДП машины AT

используется восьмибитовый номер страницы и 16-битовое смещение. Например, для адреса 23456 номер страницы - 2, смещение - 3456.

Для программирования канала 2 КПДП программа должна сначала вывести младший байт смещения в порт с адресом **4**, затем вывести в этот же порт старший байт смещения и, наконец, вывести байт номера страницы в порт с адресом **81h**.

Длина передаваемых данных выводится аналогично в порт с адресом **5** - сначала младший байт длины, затем старший.

После определения режима работы канала, адреса буфера и длины передаваемых данных, программа должна разрешить работу КПДП, выдав в порт с адресом **0Ch** байт **2**. Теперь канал прямого доступа готов к работе и будет ждать данных от контроллера НГМД.

Приведенная ниже демонстрационная программа использует несколько наиболее характерных команд контроллера НГМД. Она предназначена для работы на машине AT. Для того, чтобы она правильно работала и на машинах PC/XT, ее надо немного изменить. Изменения касаются программирования контроллера ПДП и программирования скорости передачи контроллера НГМД.

Контроллер КПДП PC/XT использует 4-битовый номер страницы буфера вместо 8-битового. Скорость передачи контроллера НГМД в машинах PC/XT не программируется, вам надо убрать соответствующие строки из программы. Еще надо обратить внимание на различное быстродействие машин AT и PC/XT и скорректировать константы в строках программы, выполняющих задержку.

Программа не проверяет, установлен ли флоппи-диск в приемный карман дисковода, поэтому перед запуском не забудьте установить диск.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include "sysp.h"

#define CYL 0

void main(void);
void fdc_out(unsigned char byte);
int  fdc_inp(void);
void int_wait(void);
void dma_init(char *);

void main(void) {

        unsigned i;
        long l;
        char buffer[512];
        char status[7], main_status;
        DPT _far *fdpt;
        FILE *sect;

        printf("\n"
                 "\nРабота с контроллером НГМД"
                 "\n  (C)Фролов А., 1991"
                 "\n");


// Эта программа предназначена только для IBM AT

        if(pc_model() != 0xfc) {
                printf("Эта программа предназначена только для IBM AT\n");
                exit(-1);
        }
```

```
// Открываем файл, в который будем записывать
// содержимое самого первого сектора на дискете

        sect = fopen("!sector.dat","wb+");

// Устанавливаем указатель на таблицу
// параметров дискеты

        fdpt = get_dpt();

// Включаем мотор дисковода A:
// Перед этим разрешаем прерывания

     _enable();
        outp(0x3F2, 0x1C);

// Выполняем задержку для разгона двигателя

        for(l=0;l<200000;l++);

// Показываем содержимое регистра основного
// состояния контроллера

        printf("Мотор включен.\t\t");
        printf("Основное состояние: %02.2X\n",inp(0x3F4));

// Перед чтением сектора необходимо установить
// головку на нужную дорожку, в нашем случае это
// дорожка с номером CYL.

// Выдаем контроллеру команду "Поиск"

        fdc_out(0xf);

// Для команды "Поиск" требуется два байта параметров:
// номер головки/номер накопителя и номер дорожки.
// Мы работаем с нулевой головкой накопителя A:,
// поэтому первый параметр равен 0, второй - CYL

        fdc_out(0);
        fdc_out(CYL);

// Показываем содержимое регистра основного
// состояния контроллера

        printf("\n<<<Поиск>>> \t\t");
        printf("Основное состояние: %02.2X\n",inp(0x3F4));

// Ожидаем прерывание по завершению операции

        int_wait();

// Задержка для позиционирования головки

        for(l=0;l<20000;l++);

// Для проверки результата выполнения команды
// "Поиск" выдаем контроллеру команду
// "Чтение состояния прерывания"

// Выводим содержимое регистра состояния
// ST0  и номер дорожки после выполнения команды
// "Поиск" PCN

        fdc_out(0x8);
        printf("Состояние прерывания:\t");
        printf(" ST0: %02.2X, \t", fdc_inp());
        printf("PCN: %02.2X\n", fdc_inp());

// Для более глубокой диагностики состояния
// контроллера выдаем контроллеру команду
```

```
// "Чтение состояния накопителя", выводим
// содержимое регистра состояния ST3

        fdc_out(4);
        fdc_out(0);
        printf("Состояние накопителя:\t ST3: %02.2X\n",fdc_inp());

// Устанавливаем скорость передачи данных 500 Кбайтов/с,
// это значение может различаться для разных типов дискет

        outp(0x3F7, 0);

// Инициализация канала прямого
// доступа к памяти

        dma_init(buffer);

// Выдаем команду "Чтение данных"

        fdc_out(0x66);
        fdc_out(0x0);      // накопитель 0, головка 0

        fdc_out(CYL);      // цилиндр CYL
        fdc_out(0);        // головка 0
        fdc_out(1);        // номер сектора - 1

// Передаем контроллеру технические параметры
// дисковода, берем их из таблицы параметров дискеты.
// Это такие параметры:
//    - размер сектора;
//    - номер последнего сектора на дорожке;
//    - размер промежутка;
//    - число считываемых/записываемых байтов

        fdc_out(fdpt->sec_size);
        fdc_out(fdpt->eot);
        fdc_out(fdpt->gap_rw);
        fdc_out(fdpt->dtl);

// Ожидаем прерывание по завершению операции

        int_wait();

// Считываем и выводим на экран байты результата
// операции "Чтение данных"

        printf("\n<<<Чтение сектора>>> \n");
        printf("   Байты состояния (ST0,ST1,ST2,C,H,R,N):\n");

        for(i=0; i<7; i++) printf("%02.2X\t", (char) fdc_inp());
        printf("\n");

// Выводим содержимое считанного сектора в файл

        for(i=0; i<512; i++) fputc(buffer[i],sect);
        fclose(sect);

// Выключаем мотор

        outp(0x3F2, 0xC);
}


// Вывод байта в контроллер дисковода

void fdc_out(unsigned char parm) {

        _asm {
                mov   dx,3F4h     // Порт основного состояния
loop_fdc_out:

                in    al,dx
```

```
                test  al,80h       // Проверяем готовность
                jz loop_fdc_out   //   контроллера

                inc   dx          // Выводим байт в порт данных
                mov   al, parm    //   контроллера
                out   dx, al
        }
}

// Ввод байта из порта данных контроллера дисковода

int fdc_inp(void) {

        _asm {
                mov   dx,3F4h     // Порт основного состояния
loop_fdc_inp:
                in    al,dx
                test  al,80h      // Проверяем готовность
                jz loop_fdc_inp   //   контроллера

                inc   dx          // Введенный байт записываем
                in    al, dx      // в регистр AX
        }
}

// Ожидание прерывания от контроллера

void int_wait(void) {

// Разрешаем прерывания

        _enable();
        _asm {
                mov   ax,40h          // После прихода прерывания
                mov   es,ax           // программа обработки прерывания
                mov   bx,3Eh          // устанавливает в 1 старший бит
wait_loop:                      // байта в области данных BIOS
                mov   dl,es:[bx]      // по адресу 0040:003E.
                test  dl,80h          // Мы ждем, когда этот бит будет
                jz    wait_loop       // установлен в 1, а затем
                                                // сбрасываем его.
                and   dl,01111111b
                mov   es:[bx],dl
        }
}

// Инициализация канала прямого доступа к памяти

void dma_init(char *buf) {

        unsigned long f_adr;
        unsigned sg, of;

// Вычисляем 24-разрядный адрес буфера для данных

        f_adr = ((unsigned long)_psp << 4)
                                + (((unsigned long)buf) & 0xffff);

// Расщепляем адрес на номер страницы
// и смещение

        sg = (f_adr >> 16) & 0xff;
        of = f_adr & 0xffff;

// На время программирования контроллера прямого
// доступа запрещаем прерывания

        _disable();

        _asm {
                mov   al,46h   // Команда чтения данных от
                                        // контроллера НГМД.
```

```
                out   12,al    // Сброс триггера-указателя байта
                                // для работы с 16-разрядными портами.
                                // Следующий байт, выводимый в 16-разрядный
                                // порт будет интерпретироваться
                                // как младший.

                out   11,al    // Установка режима контроллера ПДП

                mov   ax,of    // Смещение буфера, младший байт
                out   4,al
                mov   al,ah    // Смещение буфера, старший байт
                out   4,al

                mov   ax,sg    // Номер страницы
                out   81h,al

                mov   ax,511   // Длина передаваемых данных
                out   5,al
                mov   al,ah
                out   5,al

                mov   al,2     // Разблокировка канала 2 контроллера ПДП
                out   10,al
        }

// Инициализация контроллера закончена,
// разрешаем прерывания.

        _enable();
}
```

Остальные команды вы можете попробовать сами. Для получения дополнительной информации по контроллеру НГМД обратитесь к техническому руководству по IBM PC. Многое можно почерпнуть из описания микросхем дискового контроллера 765 фирмы NEC и аналогов этой микросхемы - Intel 8272A и отечественной КР1810ВГ72А.

1.5. Функции BIOS для работы с дисками

Наилучшим и самым безопасным способом работы с дисками на физическом уровне является использование функций BIOS. Эти функции учитывают все особенности аппаратуры и предоставляют достаточно широкий набор средств доступа к дискам на физическом уровне.

Вся дисковая подсистема обслуживается прерыванием BIOS **INT 13h**. Это прерывание выполняет множество функций. Для вызова определенной функции программа должна занести ее код в регистр **AH**, другие регистры, как правило, должны содержать параметры - номера используемых дисководов, цилиндров, головок, адреса таблиц параметров дискеты и жесткого диска и т.д.

Библиотека транслятора Microsoft QC 2.5 содержит специальную функцию **_bios_disk(),** сильно упрощающую работу с дисковыми функциями BIOS. В примерах программ, приведенных в книге, мы продемонстрируем как непосредственный вызов прерывания **INT 13h,** так и использование функции **_bios_disk()**.

Мы приведем краткую таблицу функций прерывания **INT 13h,** после чего займемся детальным описанием этих функций. В примечании к описанию функций мы будем указывать типы компьютеров, на которых данная функция работоспособна.

00h	Сброс дисковой подсистемы
01h	Получить состояние дисковой подсистемы
02h	Чтение сектора

03h	Запись сектора
04h	Проверка сектора
05h	Форматирование дорожки
06h	Форматирование дорожки (НМД)
07h	Форматирование диска (НМД)
08h	Получить текущие параметры дисковода (НМД)
09h	Инициализация таблиц параметров жесткого диска
0Ah	Чтение длинное (НМД)
0Bh	Запись длинная (НМД)
0Ch	Поиск цилиндра (НМД)
0Dh	Альтернативный сброс дисковода (НМД)
0Eh	Чтение буфера сектора (НМД)
0Fh	Запись буфера сектора (НМД)
10h	Проверка готовности дисковода (НМД)
11h	Рекалибровка дисковода (НМД)
12h	Проверка памяти контроллера (НМД)
13h	Проверка дисковода (НМД)
14h	Проверка контроллера (НМД)
15h	Получить тип дисковода
16h	Проверка замены диска
17h	Установка типа дискеты
18h	Установка среды носителя данных для форматирования
19h	Парковка головок (НМД)
1Ah	Форматирование диска (ESDI НМД)

1.5.1 Сброс дисковой подсистемы

На входе:	AH = 00h
	DL = Адрес дисковода (0, 1, ...,80h, 81h, ...)
На выходе:	-
Примечание:	PC, XT, AT, PS/2

Эта функция вызывает сброс и рекалибровку дискового контроллера (головки устанавливаются на нулевой цилиндр). Если в адресе дисковода старший бит (бит 7) установлен в 1, выполняется сброс контроллера НМД. Сброс рекомендуется выполнять после того, как произошла ошибка при выполнении других операций, таких как чтение или запись. После сброса можно попытаться повторить операцию.

Адрес дисковода 0 соответствует первому флоппи-диску (A:), 1 - второму (B:) и т.д. Адреса 80, 81 соответствуют первому и второму физическим накопителям на жестком магнитном диске.

1.5.2 Получить состояние дисковой подсистемы

На входе:	AH = 01

	DL = Адрес дисковода (0, 1, ...,80h, 81h, ...)
На выходе:	AL = Состояние дисковода после завершения последней операции
Примечание:	PC, XT, AT, PS/2

Эта функция может быть использована для анализа результата выполнения дисковой операции и получения кода ошибки. Передаваемый в регистре AL код ошибки функция берет из области данных BIOS - из байта с адресом 0000:0441h.

Код ошибки может принимать следующие значения:

00h	Успешное завершение операции
01h	Неправильная команда
02h	Не найдена адресная метка
03h	Попытка записи на диск, защищенный от записи
04h	Сектор не найден
05h	Ошибка при сбросе (НМД)
06h	Произошла замена дискеты
07h	Неправильные параметры дисковода (НМД)
08h	Переполнение канала ПДП (НГМД)
09h	Переход за границу 64К при работе с ПДП
0Ah	Обнаружен плохой сектор (НМД)
0Bh	Обнаружена плохая дорожка (НМД)
0Ch	Неправильный номер дорожки
0Dh	Неправильный номер сектора при форматировании (НМД)
0Eh	Обнаружена адресная метка управляющих данных (НМД)
0Fh	Ошибка ПДП (НМД)
10h	Обнаружена ошибка в CRC/ECC
11h	Данные скорректированы с использованием ECC (НМД)
20h	Сбой контроллера
40h	Сбой при поиске дорожки
80h	Таймаут - программа не успевает обрабатывать данные
AAh	Дисковод не готов (НМД)
BBh	Неизвестная ошибка (НМД)
CCh	Сбой при записи (НМД)
E0h	Ошибка регистра состояния (НМД)
FFh	Ошибка операции считывания (НМД)

1.5.3 Чтение сектора

На входе:	AH = 02h
	AL = Количество секторов, которые нужно прочитать

	CH = Номер дорожки
	CL = Номер сектора
	DH = Номер головки
	DL = Адрес дисковода (0, 1, ...,80h, 81h, ...)
	ES:BX = Адрес буфера для данных
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT, AT, PS/2

Эта функция позволяет прочитать один или несколько секторов диска в буфер, находящийся в оперативной памяти. Вам надо задать для начального сектора номера дорожки, головки и номер самого сектора.

Для НМД номер дорожки и сектора задаются следующим образом: биты регистра CX 5...0 задают номер сектора, а биты 15...6 - номер дорожки.

Перед чтением необходимо подготовить таблицу параметров дискеты или диска (для операций с НМД).

1.5.4 Запись сектора

На входе:	AH = 03h
	AL = Количество секторов, которые нужно прочитать
	CH = Номер дорожки
	CL = Номер сектора
	DH = Номер головки
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
	ES:BX = Адрес буфера для данных
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT, AT, PS/2

Функция записи секторов аналогична предыдущей, за исключением направления перемещения данных - данные записываются из буфера в сектора диска. Необходимо отметить, что при работе с НГМД не всякий BIOS будет ожидать разгона двигателя до рабочей скорости перед выполнением операции записи. В результате программа может получить признак ошибки. Прежде чем делать вывод о причинах ошибки, следует сбросить контроллер НГМД функцией 00H и повторить операцию записи три раза.

1.5.5 Проверка сектора

На входе:	AH = 04h

	AL = Количество секторов, которые нужно проверить
	CH = Номер дорожки
	CL = Номер сектора
	DH = Номер головки
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
На выходе:	AH = Состояние дисковода после завершения последней операции
	AL = Число проверенных секторов
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT, AT, PS/2

С помощью этой функции можно убедиться, что указанные сектора существуют и их можно прочесть. Данные проверяются по методу циклического избыточного контроля (CRC). Адрес буфера не нужен, так как чтения данных в оперативную память при проверке секторов не происходит.

Если вы используете компьютер со старой BIOS, выпущенной ранее 11/15/85, регистры ES:BX должны указывать на буфер соответствующего размера, как и при выполнении операции чтения.

Перед использованием этой функции убедитесь, что мотор НГМД раскрутился до рабочей скорости, в противном случае вы получите признак ошибки.

1.5.6 Форматирование дорожки

На входе:	AH = 05h
	AL = Количество секторов, которые нужно создать на дорожке, или Фактор чередования для НМД XT
	CH = Номер дорожки
	CL = Номер сектора
	DH = Номер головки
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
	ES:BX = Адрес буфера формата, используется для НГМД и НМД машин XT
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT, AT, PS/2

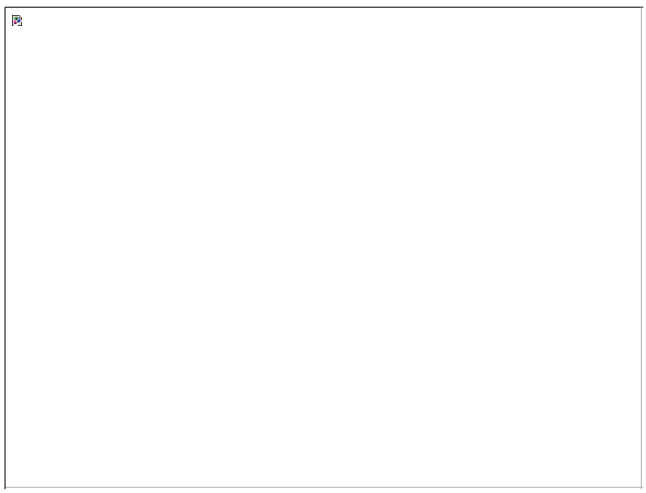
Функция форматирования предназначена для начального формирования структуры дорожки диска, она разрушает все имеющиеся на дорожке данные. С помощью функции 05 вы можете за один раз отформатировать только одну дорожку с указанным номером.

Для этой функции необходимо задать два интересных параметра, на которых мы остановимся подробнее - фактор чередования и буфер формата.

Что такое фактор чередования (Interleave)?

Этот фактор определяет последовательность расположения секторов на дорожке. Сектора могут располагаться в порядке своих номеров, через один, через два и т.д. Способ размещения секторов определяется значением фактора чередования.

Фактор 1 означает последовательное расположение секторов на дорожке в порядке их номеров, т.е. чередование отсутствует. Фактор 2 задает расположение секторов через один, 3 - через два и т.д. На рисунке показано использование фактора чередования при форматировании дорожки:



Все утилиты, предназначенные для подготовки жесткого диска к работе, требуют задания величины фактора чередования при выполнении низкоуровневого форматирования.

Для чего может понадобиться несмежное расположение секторов с последовательными номерами на дорожке диска?

При последовательном расположении секторов может получиться так, что процессор не будет успевать обрабатывать смежные сектора за один проход дорожки. Например, программа считывает последовательно второй и третий сектор. В момент времени, когда второй сектор уже считан, при быстром вращении диска к моменту начала чтения третьего сектора головки могут оказаться в середине третьего сектора и диск совершит еще один оборот, прежде чем головки окажутся в начале третьего сектора. Поэтому если программа последовательно обращается к смежным секторам, может получиться так, что при чтении каждого сектора диск будет совершать один оборот.

Если же сектора будут расположены через один или через два, количество оборотов диска, нужных для обработки последовательности смежных секторов, будет значительно меньше.

Для подбора оптимального фактора чередования можно использовать специальные программы или делать это методом проб и ошибок, задавая каждый раз новое значение фактора и проверяя быстродействие диска.

Займемся теперь буфером формата.

Перед вызовом функции форматирования регистры ES:BX должны содержать полный адрес буфера формата. Для дискет перед форматированием этот буфер должен представлять из себя заполненный массив четырехбайтовых элементов - номера дорожки, головки, сектора и кода размера сектора. Код размера сектора может иметь следующие значения:

0	128 байтов на сектор
1	256 байтов на сектор

2	512 байтов на сектор
3	1024 байтов на сектор

Количество элементов в массиве должно быть равно количеству создаваемых на дорожке секторов, то есть для каждого сектора буфер формата должен содержать один описывающий его четырехбайтовый элемент.

Для жесткого диска буфер формата должен представлять из себя массив размером 512 байтов. В начале этого массива для каждого сектора на дорожке необходимо подготовить двухбайтовые элементы. Первый байт содержит признак - хороший это сектор (00) или плохой (80h). Второй байт - это номер сектора.

Задавая последовательность номеров в буфере формата соответствующим образом, программа определяет фактор чередования.

Приведем пример подготовленного буфера формата для форматирования дорожки на 17 секторов с фактором чередования, равным 2:

```
db 00h,01h,00h,0ah,00h,02h,00h,0bh,00h,03h,00h,0ch
db 00h,04h,00h,0dh,00h,05h,00h,0eh,00h,06h,00h,0fh
db 00h,07h,00h,10h,00h,08h,00h,11h,00h,09h
```

Отметим, что буфер формата используется только для машин AT. Машины XT при форматировании НМД не используют буфер формата, вместо этого значение фактора чередования указывается при вызове функции форматирования в регистре AL.

При форматировании флоппи-дисков с помощью этой функции таблица параметров дискеты должна содержать правильное значение количества секторов на дорожке и другие параметры.

1.5.7 Форматирование дорожки (НМД)

На входе:	AH = 06h
	AL = Фактор чередования
	CH = Номер дорожки
	CL = Номер сектора
	DH = Номер головки
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
	ES:BX = Адрес буфера формата
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT

Функция форматирования дорожки с кодом 6 предназначена только для НМД и устанавливает флаг плохого сектора. Буфер формата подготавливается аналогично функции 5.

1.5.8 Форматирование диска (НМД)

На входе:	AH = 07h

	AL = Фактор чередования (только для XT)
	CH = Номер дорожки
	CL = Номер сектора
	DH = Номер головки
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
	ES:BX = Адрес буфера формата
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT

Функция форматирования с кодом 7 предназначена для форматирования целого диска начиная с определенной дорожки. Буфер формата подготавливается аналогично функции 5.

1.5.9 Получить текущие параметры дисковода (НМД)

На входе:	AH = 08h
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
	BL = тип дисковода (только для AT и PS2)
	DL = количество НМД, обслуживаемых первым контроллером
	DH = максимальный номер головки
	CL = максимальный номер сектора
	CH = максимальный номер цилиндра
	ES:DI = адрес таблицы параметров дисковода
Примечание:	PC, XT, AT, PS/2

С помощью этой функции программа может определить тип дисковода, количество дисководов, обслуживаемых первым дисковым контроллером и другие параметры дисковода, которые нужны программе для организации доступа к диску на физическом уровне. Тип дисковода, возвращаемый в регистре BL, может принимать следующие значения:

0	не используется;
1	360К, 40 дорожек, 5,25 дюймов;
2	1,2М, 80 дорожек, 5,25 дюймов;
3	720 К, 80 дорожек, 3,5 дюйма;
4	1,44М, 80 дорожек, 3,5 дюйма.

1.5.10 Инициализация контроллера НМД

На входе:	AH = 09h
	DL = Адрес дисковода (80h, 81h, ...)
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT, AT, PS/2

Функцию инициализации контроллера НМД применяют после модификации таблиц параметров жесткого диска. BIOS BIOSузнает о внесенных в таблицы изменениях и инициализирует соответствующим образом контроллер НМД.

1.5.11 Чтение секторов длинное (НМД)

На входе:	AH = 0Ah
	AL = Количество секторов, которые нужно прочитать
	CH = Номер дорожки
	CL = Номер сектора
	DH = Номер головки
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
	ES:BX = Адрес буфера для данных
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT, AT, PS/2

Функция "Чтение секторов длинное" отличается от обычной функции чтения (код 02h) тем, что она дополнительно считывает в буфер данных 4 байта кода коррекции ошибки (ECC).

1.5.12 Запись секторов длинная (НМД)

На входе:	AH = 0Bh
	AL = Количество секторов, которые нужно записать
	CH = Номер дорожки
	CL = Номер сектора
	DH = Номер головки
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
	ES:BX = Адрес буфера для данных
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT, AT, PS/2

Функция "Запись секторов длинная" отличается от обычной функции записи (код 03h) тем, что она дополнительно записывает на диск из буфера данных 4 байта кода коррекции ошибки (ECC).

1.5.13 Поиск дорожки (НМД)

На входе:	AH = 0Ch
	CH = Номер дорожки
	CL = Номер сектора
	DH = Номер головки
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT, AT, PS/2

С помощью функции 0Ch программа может подвести головки к дорожке с заданным номером. Функции чтения/записи секторов не требуют предварительного поиска дорожки, они выполняют поиск сами.

1.5.14 Альтернативный сброс дисковода (НМД)

На входе:	AH = 0Dh
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT, AT, PS/2

Вы можете использовать эту функцию для сброса контроллера вместо функции с кодом 00h. В отличие от функции сброса дисковой подсистемы с кодом 00h эта функция не влияет на контроллер НГМД, она сбрасывает только контроллер накопителя на жестком магнитном диске.

1.5.15 Чтение буфера сектора (НМД)

На входе:	AH = 0Eh
	AL = Количество секторов, которые нужно прочитать
	CH = Номер дорожки
	CL = Номер сектора
	DH = Номер головки
	ES:BX = Адрес буфера для данных
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
На выходе:	AH = Состояние дисковода после завершения последней операции

	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT

Контроллеры НМД в компьютерах PC и XT содержат внутренний буфер данных. С помощью функции 0Eh программа может прочитать содержимое этого буфера в оперативную память. Чтения данных с диска при этом не происходит. В основном функция чтения буфера используется для диагностики дискового контроллера.

1.5.16 Запись буфера сектора (НМД)

На входе:	AH = 0Fh
	AL = Количество секторов, которые нужно записать
	CH = Номер дорожки
	CL = Номер сектора
	DH = Номер головки
	ES:BX = Адрес буфера для данных
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT

Функция полностью аналогична предыдущей, за исключением того, что происходит не чтение, а запись данных из оперативной памяти в буфер контроллера. Она может быть использована для инициализации содержимого буфера сектора перед форматированием диска функцией 05h прерывания INT 13h.

1.5.17 Получить состояние дисковода (НМД)

На входе:	AH = 10h
	DL = Адрес дисковода (80h, 81h, ...)
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT, AT, PS/2

О готовности дисковода можно судить по байту состояния, передаваемому функцией в регистре AH. Этот байт аналогичен возвращаемому в регистре AH функцией 01h.

1.5.18 Рекалибровка дисковода (НМД)

На входе:	AH = 11h
	DL = Адрес дисковода (80h, 81h, ...)

На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT, AT, PS/2

Вызов функции приводит к позиционированию головок выбранного дисковода на нулевую дорожку. Дополнительно в регистре AH возвращается байт состояния дисковода.

1.5.19 Проверка памяти контроллера (НМД)

На входе:	AH = 12h
	DL = Адрес дисковода (80h, 81h, ...)
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT

Эта функция предназначена для запуска встроенной диагностики дискового контроллера, она проверяет внутренний буфер сектора и возвращает байт состояния.

1.5.20 Проверка дисковода (НМД)

На входе:	AH = 13h
	DL = Адрес дисковода (80h, 81h, ...)
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT

Функция используется для запуска внутренней диагностики контроллера.

1.5.21 Проверка контроллера (НМД)

На входе:	AH = 14h
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PC, XT, AT, PS/2

Функция запускает внутреннюю диагностику контроллера.

1.5.22 Получить тип дисковода

На входе:	AH = 15h
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
На выходе:	AH = Тип дисковода
	CX:DX = количество секторов размером 512 байтов
Примечание:	AT, PS/2

Возвращаемый этой функцией тип дисковода может принимать следующие значения:

0	диск отсутствует;
1	НГМД без аппаратных средств обнаружения замены дискеты;
2	НГМД оснащенный средствами обнаружения замены дискеты;
3	НМД.

С помощью этой функции программа может определить тип диска и возможность обнаружения замены магнитного носителя (дискеты).

1.5.23 Проверка замены диска

На входе:	AH = 16h
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
На выходе:	AH = Признак замены дискеты: 0 - замены дискеты не было; 6 - дискета была заменена.
Примечание:	AT, PS/2

В некоторых случаях замена дискеты нежелательна до выполнения определенных действий (мы говорили об этом при обсуждении драйверов дисковых устройств). С помощью этой функции программа может убедиться в том, что в дисководе установлена все та же дискета, что и в начале цикла операций. Если дискета была по ошибке заменена раньше времени, программа может потребовать установить старую дискету для завершения работы с ней.

1.5.24 Установка типа дискеты

На входе:	AH = 17h
	AL = Устанавливаемый тип
	DL = Адрес дисковода (0, 1, ..., 80h, 81h, ...)
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	AT, PS/2

Функцию установки типа дискеты необходимо использовать перед началом работы с дискетой. Устанавливаемый тип может принимать следующие значения:

0	не используется;
1	диск 360К в дисководе 360К;
2	диск 360К в дисководе 1.2М (HD);
3	диск 1.2М в дисководе 1.2М;
4	диск 720К в дисководе 720К.

Если перед вызовом этой функции был установлен флаг замены дискеты, то он сбрасывается. Дополнительно BIOS BIOSустанавливает скорость передачи данных через контроллер НГМД в зависимости от типа дискеты.

1.5.25 Установка среды для форматирования (НГМД)

На входе:	AH = 18h
	DL = Адрес дисковода (0, 1, ...)
	CH = Младшие 8 битов количества дорожек
	CL = Количество секторов на дорожку (биты 0-5)
На выходе:	AH = 00h Требуемая комбинация количества дорожек и количества секторов на дорожку поддерживается операцией форматирования;
	AH = 01h Функция недоступна;
	AH = 0Ch Функция не поддерживается или неизвестен тип дисковода;
	AH = 80h Диск не установлен в дисковод.
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	AT, PS/2

Эта функция должна быть вызвана перед использованием функции 05h форматирования диска для установки правильной скорости передачи данных через дисковый контроллер. Дополнительно функция сбрасывает флаг замены дискеты (если этот флаг установлен).

1.5.26 Парковка головок (НМД)

На входе:	AH = 19h
	DL = Адрес дисковода (80h, 81h, ...)
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PS/2

Парковка головок - это их установка в нерабочую область, т.е. на нерабочую дорожку. Эту операцию обычно выполняют перед транспортировкой компьютера для исключения повреждения дисковода.

1.5.27 Форматирование диска (ESDI НМД)

На входе:	AH = 1Ah
	AL = Количество элементов в таблице дефектов
	DL = Адрес дисковода (80h, 81h, ...)
	CL = Режим форматирования
	ES:BX = Адрес таблицы дефектов
На выходе:	AH = Состояние дисковода после завершения последней операции
	CF = 1, если произошла ошибка, 0, если ошибки нет
Примечание:	PS/2

Эта функция форматирования жесткого диска предназначена для НМД, используемого совместно с контроллером ESDI. Она поддерживает таблицу дефектных дорожек и имеет несколько режимов форматирования в зависимости от содержимого регистра CL при вызове:

Бит 0	игнорировать первичную таблицу дефектов;
Бит 1	игнорировать вторичную таблицу дефектов;
Бит 2	обновить вторичную таблицу дефектов;
Бит 3	выполнить анализ поверхности;
Бит 4	генерация периодических прерываний;
Биты 5-7	зарезервированы, должны быть равны 0.

Если при форматировании затребована функция генерации периодических прерываний, то после форматирования каждой дорожки вызывается прерывание INT 5h с регистром AH=0Fh. Это прерывание можно использовать для индикации хода процесса либо для завершения процесса форматирования по требованию оператора или программы.

При установке бита 2 регистра CL содержимое вторичной таблицы дефектов обновляется, в нее заносятся результаты тестирования диска. Для углубленного анализа поверхности диска сначала необходимо выполнить форматирование диска с битом 3, сброшенным в 0. Затем следует выполнить анализ поверхности диска, вызвав эту функцию с битом 3, установленным в 1.

1.6. Использование функций BIOS

Только что мы привели функции BIOS для работы с диском на физическом уровне. Когда и как ими пользоваться?

Доступ к диску на физическом уровне может потребоваться для чтения отдельных секторов диска, расположенных в фиксированных (или известных) местах диска - таблицы разделов диска, каталогов и т.п. С помощью функций BIOS можно выполнить низкоуровневое форматирование диска, как стандартное, так и использующее нестандартный формат дорожки.

В любом случае при записи информации в сектора следует внимательно анализировать работу программы - ошибки могут привести к разрушению логической структуры диска. В результате этого могут оказаться потеряны каталоги и файлы. Все "опасные" эксперименты лучше проводить на дискетах, и только когда вы уверены в безошибочной работе программы, можно "допустить" ее к жесткому диску.

Если вы используете дисковод с высокой плотностью записи (например, 1.2M) для работы с дискетами, использующими двойную плотность записи (360 К), перед началом работы вам надо

правильно установить скорость передачи данных через контроллер НГМД. Лучше всего это сделать функцией **17h** прерывания **INT 3h**, указав тип диска.

Не следует забывать о задержке, необходимой для разгона двигателя НГМД до рабочей скорости. Некоторые функции BIOS могут вернуть признак ошибки, если двигатель не набрал нужной скорости. Если вы получили признак ошибки, вначале следует три раза повторить вызов функции, сбрасывая каждый раз перед этим контроллер НГМД функцией 0 прерывания **INT 13h**. Если и после этого ошибка не исчезла, следует провести ее углубленный анализ.

Приведем примеры использования функций прерывания **INT 13h** для работы с НГМД.

Первый пример - программа, составленная на языке ассемблера. Она читает самый первый сектор диска, расположенный на нулевой дорожке, нулевой стороне (нулевая головка). Этот сектор имеет номер 1.

```
                .MODEL  tiny

                .STACK  100h

                .DATA

; Буфер, в который будет прочитан сектор диска

buf   db 512 dup(?)

                .CODE
                .STARTUP


                mov   ch,00h   ; Номер дорожки
                mov   cl,01h   ; Номер сектора

                mov   dh,00h   ; Номер головки (стороны диска)
                mov   dl,00h   ; Номер дисковода - дисковод A:

; Готовим адрес буфера в ES:BX

                mov   ax,cs
                mov   es,ax

                mov   bx,OFFSET buf

; Готовим код функции

                mov   ah,02h   ; Код функции - чтение сектора
                mov   al,01h   ; Количество читаемых секторов - 1

; Вызываем прерывание

                int   13h


                .EXIT   0

                END
```

Следующая программа - пример использования аппаратной поддержки проверки замены дискеты. Эта поддержка реализована в машинах класса AT, PS/2.

Сначала программа устанавливает тип дискеты. Это нужно для правильного выбора скорости передачи данных контроллером НГМД. При установке типа дискеты сбрасывается флаг замены дискеты.

Далее после чтения состояния НГМД программа делает паузу, во время которой вы можете

заменить дискету или просто открыть и закрыть дверцу дисковода. Выполнив (или не выполнив) действия по замене дискеты, нажмите на любую клавишу. Программа выведет на экран новое состояние флага замены дискеты.

Попробуйте запустить эту программу без дискеты, обратите внимание на состояние порта **0x3F7.**

Главное, что вы можете взять из приведенной ниже программы - это техника работы с флагом замены дискеты. Используя аппаратную поддержку проверки замены дискеты, ваша программа сможет более полно контролировать действия оператора по установке и замене дискет.

Текст программы:

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

union REGS inregs, outregs;

void main(void);
void main(void) {

// Устанавливаем тип диска и сбрасываем
// флаг замены дискеты

        inregs.h.ah = 0x17;
        inregs.h.al = 3;
        inregs.h.dl = 0;
        int86(0x13, &inregs, &outregs);

// Определяем тип диска и наличие аппаратной
// поддержки проверки замены дискеты

        inregs.h.ah = 0x15;
        inregs.h.dl = 0;
        int86(0x13, &inregs, &outregs);
        printf("\nТип диска A: %d",outregs.h.ah);

// Определяем состояние флага замены дискеты

        inregs.h.ah = 0x16;
        inregs.h.dl = 0;
        int86(0x13, &inregs, &outregs);
        printf("\nСостояние флага замены дискеты: %d",outregs.h.ah);

// Выводим состояние порта 0x3F7.
// Бит 7 этого порта отображает состояние
// флага замены дискеты

        printf("\nПорт 0x3F7: %02.2x",inp(0x3f7));

// Сбрасываем контроллер НГМД

        inregs.h.ah = 0;
        inregs.h.dl = 0;
        int86(0x13, &inregs, &outregs);

// Делаем паузу, во время которой можно
// заменить дискету. Запуская программу несколько
// раз, попробуйте во время ожидания нажатия на клавишу
// открыть и затем закрыть дверцу дисковода - это
// приведет к установке флага замена дискеты

        printf("\nЗамените дискету и нажмите на любую клавишу");
        getch();

// Определяем заново состояние флага замены дискеты

        inregs.h.ah = 0x16;
```

```
        inregs.h.dl = 0;
        int86(0x13, &inregs, &outregs);
        printf("\nСостояние флага замены дискеты: %d",outregs.h.ah);

// Выводим состояние порта 0x3F7.

        printf("\nПорт 0x3F7: %02.2x",inp(0x3f7));

}
```

Еще один пример - нестандартное форматирование дорожки флоппи-диска - мы приведем в следующем разделе.

1.7. Функция _bios_disk()

Стандартная библиотека трансляторов Microsoft QC .01, QC .5, C .0 содержит специальную функцию, облегчающую работу с диском на уровне BIOS - **_bios_disk()**. Эта функция требует использования файла **bios.h** и описана следующим образом:

```
unsigned _bios_disk(unsigned funct,
       struct diskinfo_t *diskinfo);
```

Параметр funct задает выполняемую функцию, параметр **diskinfo** - это указатель на структуру, описывающую необходимые параметры, такие как номер дорожки, номер головки и т.д.:

```
struct diskinfo_t
{
      unsigned drive;     // Номер дисковода
      unsigned head;      // Номер головки
      unsigned track;     // Номер дорожки
      unsigned sector;    // Номер первого сектора
      unsigned nsectors;  // Количество читаемых,
                          // записываемых
                          // или сравниваемых секторов
      void far *buffer;   // Адрес буфера в памяти
};
```

Перед использованием функции **_bios_disk()** программа должна заполнить поля структуры **diskinfo** и вызвать **_bios_disk()** с соответствующим параметром **funct**.

Файл **bios.h** содержит константы для следующих значений параметра **funct**:

_DISK_FORMAT	Форматирование дорожки, описанной параметром diskinfo функции _bios_disk(). Для этой функции программа должна задать в структуре diskinfo номер дисковода, для которого выполняется форматирование, номера головки и форматируемой дорожки. Указатель buffer программа должна установить на подготовленный буфер формата, описанный выше. Необходимо выполнить все подготовительные действия, связанные с настройкой контроллера НГМД и таблицы параметров дискеты.
_DISK_READ	Чтение одного или нескольких секторов диска. Эта функция аналогична функции 2 прерывания INT 13h. Если при чтении

	секторов произошла ошибка, ее код будет возвращен функцией _bios_disk() в старшем байте. При успешном завершении операции функция возвращает 0.
_DISK_WRITE	Запись одного или нескольких секторов на диск. Функция аналогична предыдущей, за исключением того, что данные из буфера записываются на диск.
_DISK_RESET	Сброс контроллера НГМД. Для этой функции не надо заполнять структуру diskinfo, ее содержимое игнорируется. Сброс контроллера выполняют после того, как произошла ошибка при выполнении другой операции, например, чтения или записи. После сброса можно попробовать повторить выполнение операции.
_DISK_STATUS	Получение состояния НГМД после выполнения последней операции. Старший байт возвращаемого функцией bios_disk() значения содержит байт состояния.
_DISK_VERIFY	Проверка диска. С помощью этой функции можно убедиться в том, что указанные сектора существуют и могут быть прочитаны в память. Дополнительно выполняется циклический избыточный тест (CRC). Функция проверки диска использует все поля структуры diskinfo. При ошибке старшие 8 битов возвращаемого функцией значения содержат байт состояния.

Приведем пример программы, читающей первый сектор нулевой дорожки (нулевая головка) диска **A:**. В случае ошибки программа пытается прочесть сектор три раза:

```
#include <stdio.h>
#include <conio.h>
#include <bios.h>
#include <dos.h>
#include <stdlib.h>

char _far diskbuf[512];

void main(void);

void main(void) {

        unsigned status = 0, i;
        struct diskinfo_t di;

        di.drive    = 0;
        di.head     = 0;
        di.track    = 0;
        di.sector   = 1;
        di.nsectors = 1;
        di.buffer   = diskbuf;

        for(i = 0; i < 3; i++) {
                status = _bios_disk(_DISK_READ, &di) >> 8;
                if( !status ) break;
        }
```

```
}
```

Последний пример, который мы приведем перед тем, как закончить с работой диска на физическом уровне, это форматирование дорожки. Сейчас мы будем использовать стандартное форматирование. Как отформатировать дорожку нестандартным образом, вы узнаете в разделе, посвященном защите информации от несанкционированного копирования. Там же будет приведен соответствующий пример.

Приведенная ниже программа форматирует 20-ю дорожку дискеты, установленной в дисковод A:.

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#include <bios.h>
#include "sysp.h"

// Номер форматируемой дорожки

#define TRK 20

// Код размера сектора - 512 байт

#define SEC_SIZE 2

union REGS inregs, outregs;
char _far diskbuf[512];

void main(void);
void main(void) {

        struct diskinfo_t di;
        unsigned status;
        unsigned char old_sec_size, old_fill_char, old_eot;
        int i, j;
        DPT  _far *dpt_ptr;

// Получаем адрес таблицы параметров дискеты

        dpt_ptr = get_dpt();

// Сохраняем старые значения из таблицы параметров

        old_sec_size  = dpt_ptr->sec_size;
        old_fill_char = dpt_ptr->fill_char;
        old_eot       = dpt_ptr->eot;

// Устанавливаем в таблице параметров дискеты
// код размера сектора, символ заполнения при
// форматировании, количество секторов на дорожке

        dpt_ptr->sec_size  = SEC_SIZE;
        dpt_ptr->fill_char = 0xf8;
        dpt_ptr->eot       = 15;

// Устанавливаем тип диска

        inregs.h.ah = 0x17;
        inregs.h.al = 3;
        inregs.h.dl = 0;
        int86(0x13, &inregs, &outregs);

// Устанавливаем среду для форматирования

        inregs.h.ah = 0x18;
        inregs.h.ch = TRK;
        inregs.h.cl = dpt_ptr->eot;
        inregs.h.dl = 0;
```

```
        int86(0x13, &inregs, &outregs);

// Подготавливаем параметры для функции форматирования

        di.drive    = 0;
        di.head     = 0;
        di.track    = TRK;
        di.sector   = 1;
        di.nsectors = 15;
        di.buffer   = diskbuf;

// Подготавливаем буфер формата для 15-ти секторов

        for(i=0, j=1; j<16; i += 4, j++) {
                diskbuf[i]    = TRK;
                diskbuf[i+1]  = 0;
                diskbuf[i+2]  = j;
                diskbuf[i+3]  = SEC_SIZE;
        }

// Вызываем функцию форматирования дорожки

        status = _bios_disk(_DISK_FORMAT, &di) >> 8;
        printf("\nФорматирование завершилось с кодом: %d",status);

// Восстанавливаем старые значения в
// таблице параметров дискеты

        dpt_ptr->sec_size  = old_sec_size;
        dpt_ptr->fill_char = old_fill_char;
        dpt_ptr->eot       = old_eot;

}
```