4. BIOS - пристап до тврдиот диск

Како и дискетниот диск, и тврдиот диск содржи магнетизирани плочи (дискови) што можат трајно да запишуваат податоци во вид на магнетни импулси. За разлика од дискетниот диск, тврдиот диск содржи повеќе вакви плочи. На овие плочи може да се запишува од двете страни, а тоа значи дека во тврдиот диск за секоја плоча треба да има по две глави (една на горната страна од плочата, а друга на долната страна од плочата) кои ќе вршат читање/запис на податоци.

Формат на тврдиот диск

Секој диск е поделен на ленти што од своја страна се поделени на сектори. Цилиндер претставува низа од сектори кои може да се прочитаат без да се помести главата за читање/запис при вртење на дискетните плочи. Секој цилиндар има одреден број сектори, а секој сектор има одредена големина-бајтови што може да се запишат во него.

Партиции

Партициите овозможуваат конфигурирање на делови од тврдиот диск за различни оперативни системи. Тоа се прави така што се алоцираа одреден број цилиндри за различни оперативни системи при форматирање на дискот. Првиот сектор од тврдиот диск ја содржи информацијата за оваа поделба. Оваа информација вклучува податок за почетокот на секоја партиција и нејзината големина, како и за оперативниот систем. Исто така, овој сектор содржи информација кој оперативен систем е активен и кој оперативен систем се активира при подигање на системот.

BIOS-от пристапува до тврдиот диск користејќи го прекинот 13h- истиот прекин како и за дискетната единица. Индивидуалните функции се идентични како за дискетната единица, така и за тврдиот диск, но контролата на тврдиот диск е различна во однос на контролата на дискетната единица. Имено, кога се повикува прекинот 13h, најпрво се пристапува до процедурата за тврдиот диск. Оваа процедура тестира дали се работи за тврдиот диск или за дискетната единица. Ако се работи за тврдиот диск се повикува соодветната процедура. Ако е адресирана дискетната единица (A или B) се повикува прекинот 40h, кој од своја страна покажува на стариот прекин 13h.

Сите процедури на тврдиот диск враќаат код за успешноста на своето извршување, што се сместува во регистарот AH. Индивидуалните кодови ги имаат овие значења:

01h	Адресирана функција или единица што не постои
02h	Адресата не е најдена
04h	Секторот не е најден
05h	Грешка при ресетирање на контролерот
07h	Грешка при иницијализација на контролерот
09h	Грешка при DMA трансмисија
0Ah	Дефектен сектор
10h	Грешка при читање
11h	Грешка при читање, корегирана со ECC
20h	Дефект на контролерот
40h	Прекината операција на барање
80h	Дискетната единица не постои
AAh	Дискетната единица не е спремна
CCh	Грешка при запис

Ако се појави грешка при процедурата на читање, прочитаните податоци не мора да бидат дефектни. Ова покажува дека се јавила грешка при читање, но дека таа може да биде корегирана со помош на ECC (Error Correction Code) алгоритмот. Оваа процедура е слична со CRC (Cyclic Redundancy Check) процесот во дискетните дрејвови. При записот комплицирана математичка формула додава четири бајти на секој сектор. Резултат од овој процес е добивање на сектор плус четири бајти. Ако се појави грешка при читање, тогаш таа се корегира со помош на снимените четири бајти и ECC алгоритмот. Пред повикот на прекинот 13h, се прави следново: бројот на функцијата се сместува во регистарот AH, тврдиот диск што треба да се адресира се сместува во регистарот DL (80h - за првиот тврд диск, 81h - за вториот тврд диск), бројот на главата за читање со сместува во DH регистарот, CH регистарот го содржи бројот на цилиндерот.

Функција 0H: ресетирање на дискот

Функцијата 0H од прекинот 13h, го ресетира тврдиот диск. Овој ресет се извршува автоматски при подигање на системот, но се појавува и кога ќе се појави некоја грешка при повикување на другите BIOS функции за пристап до дискетната единица. Кој диск ќе се ресетира, зависи од вредноста што се наоѓа во DL регистарот пред повикување на прекинот. Пред повикување на прекинот, вредноста 0H треба да се смести во регистарот AH. По повикување на прекинот, статусот се враќа во регистарот AH.

Функција 01H: статус

Функцијата 1 го чита статусот на тврдиот диск без негов пристап. Ако по повикот на прекинот се врати вредност 0, тоа значи дека нема грешка, но вратена вредност различна од 0 ја има следнава интерпретација:

01H	Функцискиот број не е прифатен
02H	Маркирањето на адресата не е најдено
03H	Неуспешен запис на заштитен диск
04H	Адресата на секторот не е најдена
08H	DMA грешка
09H	Пренос на податоци
10H	Грешка при читање
20H	Грешка на контролерот на дискот
40H	Лентата не е најдена
80H	Временски изминат период, дикетата не се наоѓа во диск. единица

Ако една од овие грешки се појави, операцијата над дискот ќе се повтори неколку пати проследена со ресет на дискот.

Функција 2: читање

Функцијата 02h чита еден или повеќе сектори. Обичен повик на оваа функција може да прочита до 128 сектори. Ова ограничување е поради тоа што контролерот на тврдиот диск ги пренесува податоците директно во PAM-от преку DMA. DMA може да пренесе најмногу 64 Kb во еден момент. Тоа значи дека комплетниот бафер чија адреса се наоѓа во парот ES:BX е во граници од 64 K, почнувајќи од сегментната адреса во ES. Оваа функција ги чита сите сектори по реден број.

Функција 3: запис на сектор

Функцијата 03x запишува еден или повеќе сектори. Обичен повик на оваа функција може да запише до 128 сектори.

Функција 4: проверка на дискот

Функцијата 4 тестира дали податоците се пренесени коректно кон/од дискот. Коректниот пренос на податоците се базира на CRC (Cyclical Redundancy Check) вредноста, што се заснова на сума од вредноста на секој бајт од секторот на кој подоцна се извршуваа соодветна математичка операција.

Функција 5: форматирање

Оваа функција овозможува корисникот да форматира дел од дискот. Пред повикување на прекинот, бројот на функцијата (5) мора да се смести во регистарот AH, AL регистарот го содржи бројот на сектори што треба да се форматираат, а бројот на лентите што треба да се форматираат се пренесува во регистарот CH. Бројот на дискетниот драјв се сместува во DL регистерот (C = 80h, D = 81h), страната на дискот во DL регистерот. При ова форматирање се користат информации од 11 бајтната табела DDPT опишана подолу.

DDPT (Disc Drive Parametar Table)

За програмирање на дискетниот контролер, BIOS-от треба да ги знае физичките информации за форматирање. ROM BIOS-от содржи табела за секоја дискетна единица. Оваа табела може да ја дефинира и самиот корисник, а BIOS-от се референцира на оваа табела преку FAR поинтери (покажувачи).

DOS-от креира своја DDPT, со што се зглемува брзината на пристапот до дискетната единица. DDPT е со должина од 11 бајта, и кога корисникот дефинира свој DDPT, секој од параметрите не може да се промени. Во табелата е прикажана структурата на DDPT (полињата со * корисникот може да ги менува):

Офсет	Значење	Тип
*00H	големина на чекорот и подигање на главата	1 Byte
*01H	полнење	1 Byte
*02H	време на работа при подигање на системот	1 Byte
03H	големина на секторот	1 Byte
04H	број на сектори по лента	1 Byte
05H	должина на GAP3 при читање/запис	1 Byte
06H	должина на податокот	1 Byte
07H	должина при форматирање	1 Byte
*08H	знак за полнење при форматирање	1 Byte
*09H	време на поставување на главата	1 Byte
*0AH	време на дискетниот мотор	1 Byte

Функција 08H: детекција на параметрите на дискот

По повикот на функцијата 08H, DL го содржи бројот на тврди дискови приклучени на контролерот на дискот (вратената вредност може да биде 0, 1 или 2 приклучени диска). DH регистарот го содржи бројот на глави за читање/запис. Овој број е релативен на нулата, т.е. ако се врати вредност 7, тоа значи дека има 8 глави за читање/запис. Бројот на цилиндри се враќа во регистарот CL (битови 0 - 7) и во двата горни бита од регистарот CH (битови 8, 9). Исто така, и овде вратениот број е релативен на нулата. Бројот на сектори по лента се сместува во ниските 6 бита на регистарот CH (бит 0 -5). Овој број е релативен на 1, а не на нула.

Тврдиот диск освен податочните бајтови, содржи и други компоненти. Кога се форматира сектор, одредена количина на бајтови се додава, за да може пода-

тоците коректно да се прочитаат/-/запишат во тој сектор (види слика долу). Секој сектор почнува со серија од 13 бајти со вредност 0 за синхронизација (SYNC field). Следното поле е ID - за идентификација на секторот. Овде се сместени цилиндерот, главата и бројот на секторите. Оваа информација почнува со специјален бајт, а завршува со двобајтен код на грешка (CRCID поле). Следното поле е GAP, што му кажува на контролерот дека треба да почне со читање на податоците. Потоа следува податочното поле (DATA). Податочното поле завршува со два бајта на корекција (ECC DATA field) што контролерот го користи за проверка на валидноста на снимените податоци.

0	13	15	17	18	20	36	38	550	552
SYNC	ID	ZK	S	CRCTP	GAP	AM	DATA	ECC	GAP

Пример:

```
/*****************************************************/
/*              F I X P A R T C . C                  */
/*    : Gi prikazuva particiite na tvrdiot dik.      */
/*      Memoriski model   : SMALL                    */
/*****************************************************/
    #include <dos.h>
    #include <string.h>
    #include <stdlib.h>
    #include <stdio.h>
/*== Constanti ======================================*/
#define TRUE  ( 1 == 1 )
#define FALSE ( 1 == 0 )
/*== Macroa =========================================*/
#define HI(x) ( *((BYTE *) (&x)+1) )     /* Go vraka visokiot bajt od zborot */
#define LO(x) ( *((BYTE *) &x) )         /* Go vraka niskiot bajt od zborot  */
typedef unsigned char BYTE;
typedef unsigned int WORD;
typedef struct {                         /* Opis na pozicija na sektorot */
                 BYTE Head;              /* Glava za citanje/zapis */
                 WORD SecCyl;            /* Sektor i broj na cilindar */
               } SECPOS;

typedef struct {
          BYTE       Status;            /* Status na partiticija */
          SECPOS     StartSec;          /* Prv sector */
          BYTE       PartTyp;           /* Tip na particija */
          SECPOS     EndSec;            /* Posleden sektor */
          unsigned long SecOfs;         /* Ofset od boot sektorot */
          unsigned long SecNum;         /* Broj na sektori */
     } PARTENTRY;

typedef struct {                        /* Go opisuva sektorot na particija */
          BYTE       BootCode[ 0x1BE ];
          PARTENTRY PartTable[ 4 ];
          WORD       IdCode;            /* 0xAA55 */
     } PARTSEC;

typedef PARTSEC far *PARSPTR;           /* pokazuvac na sektorot na */
                                        /* particija vo memorijata  */

/*****************************************************************/
/*  ReadPartSec : Cita sektor od tvrdiot disk.                   */
/*  Input: - DS       : BIOS cod za drajvot (0x80, 0x81, .)      */
/*         - Head     : Broj na glavi za citanje/zapis           */
/*         - SctCyl   : Broj na Sector/cilinder vo BIOS format   */
/*         - Buf      : Bafer do koj sektorot se prenesuva       */
/*  Output: TRUE ako sektorot moze da se procita bez greski,     */
/*          inaku FALSE                                          */
/*****************************************************************/

BYTE ReadPartSec( BYTE DS, BYTE Head, WORD SctCyl, PARSPTR Buf )
{
union REGS  Regs;       /* Registri na procesorot za povik na prekinot */
struct SREGS SRegs;

Regs.x.ax = 0x0201;
Regs.h.dl = DS;
Regs.h.dh = Head;
Regs.x.cx = SctCyl;
Regs.x.bx = FP_OFF( Buf );
SRegs.es  = FP_SEG( Buf );

int86x( 0x13, &Regs, &Regs, &SRegs );  /* povik na prekinot na tvrdiot disk */
return !Regs.x.cflag;
}

/*****************************************************/
/*  GetSecCyl: Zema cilinder kodiran od BIOS         */
/*             broj na sektor i cilindar             */
/*  Input  : SctCyl   : Vrednost (to treba da se dekodira */
/*           Sector   : Cilindar sto treba da se referencira */
/*           Cylinder : Cilinder sto treba da se referencira */
/*  Output : Nisto                                   */
/*****************************************************/

void GetSecCyl( WORD SctCyl, int *Sector, int *Cylinder )
{
 *Sector   = SctCyl & 63;                    /* bitovi za maskiranje 6 i 7 */
 *Cilinder = HI(SctCyl) + ( ( (WORD) LO(SctCyl) & 192 ) << 2 );
}

/*****************************************************************/
/*  ShowPartition: Gi prikazuva particiite na tvrdiot disk na ekran  */
/*  Input  : DS : broj na soodvetniot tvrd disk (0, 1, etc.)     */
/*  Output : Nisto                                               */
/*****************************************************************/

void ShowPartition( BYTE LW )
{
 #define AP ( ParSec.PartTable[ Entry ] )
BYTE       Head,              /* Glava na tekovnata particija */
           Entry;             /* Brojac na jamka */
int        Sector,            /* Zemi sektor i  */
           Cylinder;          /* broj na cilinder */
PARTSEC    ParSec;            /* tekoven sektor na particijata */
union REGS Regs;              /* registri na procesorot za povik na prekinot */

printf("\n");
LW |= 0x80;
if (ReadPartSec( LW, 0, 1, &ParSec) )       /* Cita sektor od particija    */
  {                                         /* Sectorot moze da se procita */
   Regs.h.ah = 8;
   Regs.h.dl = LW;
   int86( 0x13, &Regs, &Regs );         /* Povik na prekinot na tvrdiot disk */
   GetSecCyl( Regs.x.cx, &Sector, &Cylinder );
printf( "* Drive %2d:     %2d Glavi %4d*
        * Cilindri %3d Sectori                          *\n",
        LW-0x80, Regs.h.dh+1, Cylinder, Sector );
printf( "* Partition table in Partition Sector       *\n");
printf( "*  ³    ³                    ³     Start    ³*
        *     End      ³Dist fm³       *\n");
printf( "*³Nr³Boot³Typ                ³Head Cyl. Sec ³*
        "Head Cyl. Sec ³BootSec³Total  *\n");
/*-- Zema tabela na particija ---------------------------------------*/
   for ( Entry=0; Entry < 4; ++Entry )
    {
     printf( "* %d³", Entry );
     if ( AP.Status == 0x80 )                    /* Particijata e aktivna? */
         printf(" Da");
     else
         printf (" Ne ");
     printf("³");
     switch( AP.PartTyp )                  /* Presmetaj go tipot na Particijata */
      {
       case 0x00 : printf( "Ne e alociran      " );
                   break;
       case 0x01 : printf( "DOS, 12-bit FAT    " );
                   break;
       case 0x02 :
       case 0x03 : printf( "XENIX              " );
                   break;
       case 0x04 : printf( "DOS, 16-bit FAT    " );
                   break;
       case 0x05 : printf( "DOS, ext. partition" );
                   break;
       case 0x06 : printf( "DOS 4.0 > 32 MB    " );
                   break;
       case 0xDB : printf( "Concurrent DOS     " );
                   break;
       default   : printf( "Unknown (%3d)      ",
                           ParSec.PartTable[ Entry ].PartTyp );
      }

     /*-- Zema fizicki i logicki parametri ----------------------*/
     GetSecCyl( AP.StartSec.SecCyl, &Sector, &Cylinder );
     printf( "³%2d %5d  %3d ", AP.StartSec.Head, Cylinder, Sector );
     GetSecCyl( AP.EndSec.SecCyl, &Sector, &Cylinder );
     printf( "³%2d %5d  %3d ", AP.EndSec.Head, Cylinder, Sector );
     printf( "³%6lu ³%6lu *\n", AP.SecOfs, AP.SecNum );
    }
}
 else
 printf("Greska pri pristap na boot sektorot\n");
}

/*****************************************************
*            G L A V N A    P R O G R A M A          *
*****************************************************/

int main( int argc, char *argv[] )
{
int DS;
DS = 0;
 if ( argc == 2 )             /* Ako korisnikot vnese razlicen argument? */
  {                                                          /* da */
   DS = atoi ( argv[1] );
   if ( DS == 0 && *argv[1] != '0' )
    {
     printf("\nedobar broj na disk!");
     return( 1 );                                  /* Kraj na  programata */
    }
  }
 ShowPartition( (BYTE) DS );       /* Go prikazuva sektorot na particija */
 return( 0 );
}
```

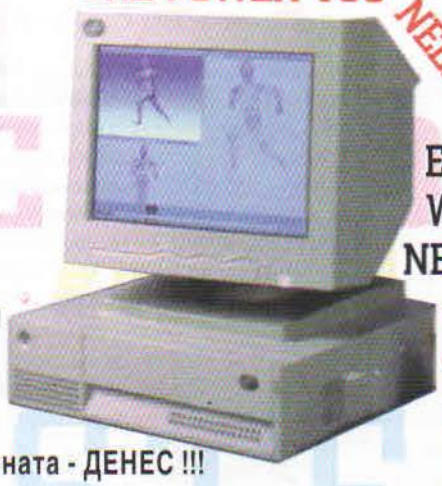
С.Г.

NETWORKING YOUR RESOURCES

FASTFORWARD

IBM & IBM COMPATIBLE

THE POWER YOU NEED

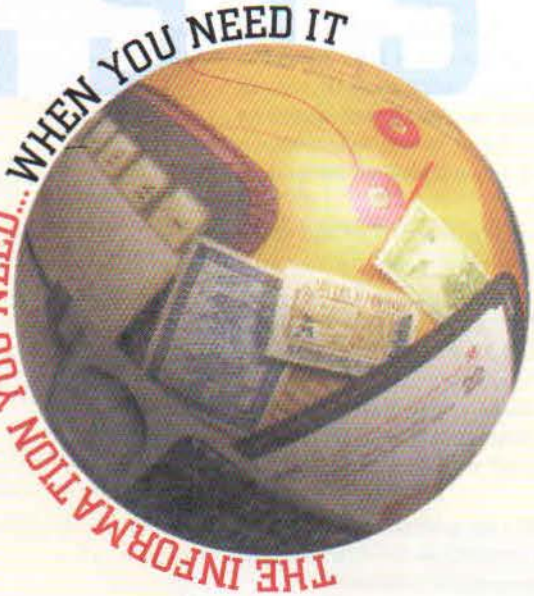


EXACTLY WHEN YOU NEED IT...

NOW !!!

POWER **PC** извонредни RISC системи кои овозможуваат БРЗИНА, МОЌ и СТАБИЛНОСТ. Една нова, динамична опција на сервери и работни станици на иднината - ДЕНЕС !!!

THE SYSTEM



NETWARE 4.1

Ако ја знаете важноста на Вашата мрежа, тогаш сигурно сте запознати со важноста на новиот NETWARE 4.1 пакет... а, ако сé уште не сте, тогаш јавете ни се, со задоволство ќе Ви кажеме сé што треба да знаете - и зошто...

APC™

FROM THE PEOPLE WHO KNOW NETWORKS BEST...

GORD SYSTEMS

IBM Business Partner

Novell Authorized Reseller

APC AMERICAN POWER CONVERSION AUTHORISED DEALER

TEN YEARS 10 '85 '95 ДЕСЕТ ГОДИНИ

РУЗВЕЛТОВА 6 / ГАЛ. 6, 91000 СКОПЈЕ, МАКЕДОНИЈА. ТЕЛЕФОН : (389 91) 233 419 / 239 118 ФАКС : (389 91) 234 805