ПРОГРАММНЫЕ МЕТОДЫ ТЕСТИРОВАНИЯ НАКОПИТЕЛЕЙ НА ЛАЗЕРНЫХ ДИСКАХ

Маргулис М.Б., Трушенко А.Н.
Кафедра ЭВМ ДГТУ

Abstract

Margulis M.B., Trushenko A.N. CD-ROM testing methods. *The CD-ROM testing is useful and interesting element of diagnostic personal computers. This article contents CD-ROM diagnostic and testing methods as C++ modules. The programm function used certain testing sequence.*

Введение

Тестирование аппаратных ресурсов компьютера включает в себя несколько основных этапов. Интересным направлением этой задачи является проверка накопителя на лазерных дисках CD-ROM. В этой статье исследованы некоторые методы тестирования накопителя и носителя информации, а также определение специфической информации о параметрах и свойствах драйвера, накопителя и дисков CD-ROM. Все примеры реализованы в виде функций на языке С (компилятор Borland C++ 3.1).

1. Определение основных параметров драйвера устройства накопителя CD-ROM

Прежде всего, необходимо определить наличие драйвера устройства CD-ROM, его внутреннюю версию, количество поддерживаемых накопителей и их логические символьные имена. Указанные задачи реализуются в отдельной функции *char CheckDriver()*, которая может распознавать три основных драйвера CD-ROM: mscdex (MicroSoftCD-romEXtension), corelcdx и lotuscd. Кроме того, определяется версия и количество накопителей (+ логические имена):

```
char CheckDriver()
  {
    unsigned tmp1,tmp2;
    asm {
          push  0DADAh              // Засылка в стек контрольного значения
          mov   AX,1100h            // Вызов функции
          int   2Fh
          pop   tmp1                // Анализ состояния стека после вызова
       }                            // прерывания
    if (tmp1==0xDADA) return 0;     // Ни один драйвер не установлен
    asm {
          mov   AX,15FFh            // Проверка, установлен ли драйвер
          sub   BX,BX               // corelcdx (замена mscdex)
          int   2Fh
          mov   tmp2,BX
          mov   AX,1500h            // Определение количества накопителей,
          sub   BX,BX               // поддерживаемых установленным
          int   2Fh                 // драйвером
          mov   NumberOfCD,BX
          mov   AX,150Ch            // Получение версии текущего драйвера.
          sub   BX,BX               // Функция возвращает старшую (BH)
          int   2Fh                 // и младшую (BL) версии драйвера
          mov   Version,BX
```

```
            mov   AX,150Dh          // Определение списка логических
            les   BX,DriverLetters  // дисков CD-ROM в системе
            int   2Fh
        }
    if (tmp2==0xABCD) return 4;     // Установлен драйвер corelcdx
    if (tmp1==0xADAD) return 1;     // Установлен драйвер mscdex
    if (tmp1==0xDADB) return 2;     // Установлен lotus CD/Networker
    return 8;                       // Установлен другой (совместимый с
}                                   // mscdex) драйвер устройства
```

После проверки наличия установленного драйвера можно получить более детальную информацию о драйвере накопителя на лазерных дисках. Эта функция реализуется на основе определения адреса, по которому драйвер расположен в памяти:

```
void GetDrvDeviceList()
  {
    unsigned char *Buffer;
    if ((Buffer=(char *)malloc(5))==NULL) PrintErrDOS(8,1);
asm {
            mov   AX,1501h          // Занесение номера функции
            les   BX,Buffer         // Установка адреса ES:BX на массив
            int   2Fh
    }
    SubUnit=Buffer[0];                      // Чтение номера устройства
    DrvSeg=(Buffer[4]<<8)+Buffer[3];        // Вычисление сегмента и
    DrvOffs=(Buffer[2]<<8)+Buffer[1];       // смещения драйвера
    free(Buffer);                           // Освобождение занимаемой памяти
  }
```

Зная адрес драйвера (*DrvSeg:DrvOffs*) можно легко считать любую информацию, сообщаемую драйвером. К примеру, зная, что символьное имя устройства находится по смещению 0Ah можно осуществить вывод этого имени на экран следующим образом:

```
for (i=10;i<18;i++) printf("%c",peekb(DrvSeg,DrvOffs+i));
```

2. Определение типа файловой системы на носителе CD-ROM

Существуют различные стандарты и типы файловых систем, применяемых для записи информации на диски CD-ROM, например: High-Sierra (CDROM), ISO 9660 (CD001), стандарт музыкального диска (AudioCD) и т.д. Определить тип файловой системы, а также серийный номер и метку диска можно следующим образом:

```
void GetFileSystem(char DiskNum)
  {
    unsigned char *SysType; unsigned Error,Flags;
    if ((SysType=(char *)malloc(25))==NULL) PrintErrDOS(8,1);
    asm {
          push  DS
          mov   AX,6900h
          sub   BX,BX
          mov   BL,DiskNum          // Занесение номера логического
          inc   BL                  // диска CD-ROM
          lds   DX,SysType
          int   21h
          pushf                     // Сохранение регистра флагов
          pop   Flags               // для проверки правильности операции
          mov   Error,AX            // Сохранение кода ошибки
          pop   DS
        }
    if (Flags&1) PrintErrDOS(Error,0);
    SerNumHi=(Buffer[5]<<8)+Buffer[4];    // Вычисление серийного номера
    SerNumLo=(Buffer[3]<<8)+Buffer[2];    // читаемого диска
```

```
    for (i=6;i<17;i++)                    // Чтение метки тома
       Label[i-6]=SysType[i];
    for (i=17;i<25;i++)                   // Чтение имени файловой системы
       SysType[i-17]=SysType[i];          // (CDROM, CD001 или AudioCD)
    free(SysType);
}
```

3. Определение текущего состояния накопителя

Одним из главных требований, предъявляемых к любому устройству – это его совместимость со стандартами. Чем больше количество возможностей поддержки встроено в устройство, тем оно считается более эффективным. Конфигурация состояния CD-ROM описывается несколькими параметрами:

1. Определение поддерживаемых форматов пересылки секторов: Raw Reading – режим "сырого" чтения (возможность читать полный сектор с EDC и ECC (Error Detection and Error Correction Code), Cooked – режим "приготовленного" чтения (аппаратная обработка ошибок и возврат чистых данных);
2. Состояние дверцы привода (откр/закр);
3. Возможность записи информации на носитель;
4. Способность устройства проигрывать аудио/видео дорожки;
5. Поддержка прямого контроля аудио канала;
6. Поддерживаемые стандарты адресации (Red Book или HSG);
7. Наличие диска в дисководе и т.д.

Описанные выше свойства можно определить в разработанной функции, которая вызывается с параметром Mode0=6. Кроме того, задавая различные режимы параметром Mode0, можно определить:

- Mode0=7 Размер сектора на диске. Обычно это 2048 байта в режиме Cooked (чистые данные) и 2352 байта в режиме Raw (чистые данные + коды EDC, ECC + служебные данные);

- Mode0=8 Размер диска в секторах. Отсюда можно вычислить общее количество байт данных на носителе по формуле:

$$РазмерДиска = (РазмерСектора)*(ОбщееЧислоСекторов)/1048576\ [Mb];$$

- Mode0=10 Получение информации об аудио дорожках – номер и стартовый адрес первой звуковой дорожки (Сектор HSG) и номер последней дорожки. Сектор HSG – это один из форматов представления адреса. Пересчет временных параметров в адрес можно осуществить по следующей формуле:

$$СекторHSG = Минута * 4500 + Секунда * 75 + Фрейм - 150,$$

где Фрейм – это звуковой квант, равный 1/75 секунды звучания;

- Mode0=11 Выдача специфической информации о дорожке – стартовый адрес и признак дорожки (Табл. 1). Различают информационные дорожки и музыкальные. На дисках стандарта Yellow Book (Mode 1. HS или ISO 9660). могут встречаться и данные и музыкальные дорожки. В этом случае все цифровые данные будут интерпретированы как дорожка #1, а все последующие музыкальные треки – соответственно дорожки #2, #3 и т.д.;

- Mode0=12 Во время проигрывания аудио информации функция возвращает номер текущей дорожки, а также сведения о положении лазера относительно начала дорожки и начала всего диска. Данные возвращаются в формате Red Book (4-байтная структура – 1-й байт – минута, 2-й байт – секунда, 3-й байт – фрейм).

Табл. 1 Значение слова со служебными данными дорожки (x – любое значение)

Значение	*Аудиоканалы*	*Предварительное усиление*	*Защита от копирования*
0xxx	2	Нет	Нет
1xxx	2	Есть	Нет
2xxx	2	Нет	Есть
3xxx	2	Есть	Есть
4xxx	Данные	–	Нет
6xxx	Данные	–	Есть
8xxx	4	Нет	Нет
9xxx	4	Есть	Нет
Axxx	4	Нет	Есть
Bxxx	4	Есть	Есть

```
// --- Mode0: 06 - Device Status
//            07 - Sector Size (Mode1=0 - Cooked, Mode1=1 - Raw)
//            08 - Volume Size in Sectors
//            10 - Audio Disk Info
//            11 - Audio Track Info (Mode1=Track Number)
//            12 - Audio Q-Channel Info (Playing Status)
void IOCTL_Input(char DiskNum,char Mode0,char Mode1,char NumByte2Rd)
  {
    unsigned char *ReqHeader,*Data;  unsigned Error,Flags;
    if ((ReqHeader=(char *)malloc(20))==NULL) PrintErrDOS(8,1);
    if ((Data=(char *)malloc(NumByte2Rd))==NULL) PrintErrDOS(8,1);
    ReqHeader[0]=20;                  // Длина запроса
    ReqHeader[1]=SubUnit;             // Номер устройства
    ReqHeader[2]=3;                   // Функция DOS IOCTL Input
    Data[0]=Mode0;                    // Режимы работы драйвера
    Data[1]=Mode1;
    ReqHeader[14]=(char )FP_OFF(Data)&0x00FF;  // Занесение адреса
    ReqHeader[15]=(char )FP_OFF(Data)>>8;      // массива Data в
    ReqHeader[16]=(char )FP_SEG(Data)&0x00FF;  // структуру данных, пере-
    ReqHeader[17]=(char )FP_SEG(Data)>>8;      // даваемых драйверу
    ReqHeader[18]=NumByte2Rd;                  // Кол-во байт для чтения
    ReqHeader[19]=0;
    asm {
          mov   AX,1510h
          sub   CX,CX
          mov   CL,DiskNum          // логический диск
          les   BX,ReqHeader        // адрес запроса
          int   2Fh
          pushf
          pop   Flags
          mov   Error,AX
      }
    if (Flags&1) PrintErrDOS(Error,1);
    if (ReqHeader[4]>>7) PrintErr(ReqHeader[3],0);
    switch (Mode0)
      {
       case  6:  // Выдача статуса накопителя (2 байта Data[1]-[2])
       case  7:  // Выдача размера сектора (2 байта Data[1]-[2])
       case  8:  // Выдача размера диска в секторах (4 байта Data[1]-[4])
       case 10:  // Выдача номера первой аудио дорожки (1 байт Data[1]),
                 //            последней аудио дорожки (1 байт Data[2]),
                 //       стартового адреса первой дорожки (формат HSG)
                 //                                 (4 байта Data[3]-[6])
       case 11:  // Выдача спец. информации о дорожке
                 // Номер дорожки (1 байт Data[1])
                 // Стартовый адрес (HSG) (4 байта Data[2]-[5])
                 // Контрольная информация (1 байт Data[6]) см. Табл.1
```

```
        case 12:  // Текущая активная дорожка (1 байт Data[2])
                  // Текущее время игры дорожки
                  // (3x1 байт: мин-Data[4], сек-Data[5], фр-Data[6])
                  // Текущее время игры диска (Data [8]-[10])
      }
    free(ReqHeader);
    free(Data);
  }
```

4. Тестирование механики привода

Программным путем можно проверить работоспособность механических частей накопителя CD-ROM. В частности это: открыть/закрыть дверцу привода, позиционировать лазер на какой-либо заданный сектор диска и провести аппаратный сброс (инициализацию). Вышеперечисленные действия можно осуществить, используя разработанную функцию *void IOCTL_Output(...)*:

```
// --- Mode0: 00 - Eject Disk;  02 - Reset Drive (Init);  05 - Close Tray
void IOCTL_Output(char DiskNum,char Mode0,char NumByte2Wr)
  {
    unsigned char *ReqHeader,*Data;  unsigned Error,Flags;
    if ((ReqHeader=(char *)malloc(20))==NULL) PrintErrDOS(8,1);
    if ((Data=(char *)malloc(NumByte2Wr))==NULL) PrintErrDOS(8,1);
    ReqHeader[0]=20;                  // Количество байт для передачи
    ReqHeader[1]=SubUnit;             // Номер устройства
    ReqHeader[2]=12;                  // Номер функции IOCTL Output
    Data[0]=Mode0;                    // Режим работы
    ReqHeader[14]=(char )FP_OFF(Data)&0x00FF;  // Занесение адреса
    ReqHeader[15]=(char )FP_OFF(Data)>>8;      // массива Data в
    ReqHeader[16]=(char )FP_SEG(Data)&0x00FF;  // структуру данных, пере-
    ReqHeader[17]=(char )FP_SEG(Data)>>8;      // даваемых драйверу
    ReqHeader[18]=NumByte2Wr;                  // Кол-во байт передачи
    ReqHeader[19]=0;
    asm {
          mov   AX,1510h
          sub   CX,CX
          mov   CL,DiskNum          // логический диск
          les   BX,ReqHeader        // адрес заголовка запроса
          int   2Fh
          pushf
          pop   Flags               // Сохранение регистра флагов
          mov   Error,AX            // для анализа ошибки
        }
    if (Flags&1) PrintErrDOS(Error,1);
    if (ReqHeader[4]>>7) PrintErr(ReqHeader[3],0);
    free(ReqHeader);          // Освобождение занимаемой
    free(Data);               // памяти
  }
```

5. Проверка целостности поверхности диска

Проверка поверхности может проводиться одним и тем же способом и для музыкальных дорожек и для цифровых данных. Это посекторное считывание и проверка правильности произведенной операции. Альтернативным методом проверки музыкальных дорожек может служить проигрывание какого-либо множества секторов. При этом можно "услышать" ошибки. При попадании лазера на сбойный/нечитаемый сектор происходит его восстановление по кодам EDC и ECC. В случае неудачного восстановления (или ошибочного) можно услышать посторонний шум.

Разработанная функция *void TestAudio(...)* реализует режимы мини-проигрывателя (задается стартовый сектор и общее количество секторов для

воспроизведения). Сочетая оба способа, можно отслеживать критические ошибки (ошибка чтения сектора и воспроизведения) и некритические (ошибка чтения сектора, но восстановление по кодам музыкального ряда).

```
// Mode: 0x84 - Play;  0x85 - Stop;  0x88 - Pause
void TestAudio(char DiskNum,char Mode)
  {
     unsigned char *ReqHeader;  unsigned Error,Flags;
     if ((ReqHeader=(char *)malloc(22))==NULL) PrintErrDOS(8,1);
     ReqHeader[0]=22;
     ReqHeader[1]=SubUnit;
     ReqHeader[2]=Mode;
     ReqHeader[13]=1;
     ReqHeader[14]-[17] = стартовый сектор для проигрывания
     ReqHeader[18]-[21] = количество проигрываемых секторов HSG
     asm {
            mov   AX,1510h
            sub   CX,CX
            mov   CL,DiskNum
            les   BX,ReqHeader
            int   2Fh
            pushf
            pop   Flags
            mov   Error,AX
       }
     if (Flags&1) PrintErrDOS(Error,0);
     if (ReqHeader[4]>>7) PrintErr(ReqHeader[3],0);
     free(ReqHeader);
  }
```

Заключение

Для организации комплексной проверки состояния накопителя и дисков CD-ROM компьютера, предлагаемые функции диагностики чаще всего используются совместно. При этом, основополагающими элементами тестирования служат функции проверки поверхности накопителя CDROM, использующие прерывание 2Fh .

Литература

1. Документация "MS-DOS CD-ROM Extension". Microsoft, 1995.
2. Мюллер С. "Модернизация и ремонт персональных компьютеров".– М.: Бином, 1996.

Поступила в редакційну колегію 20.04.2000 р.